UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Aplicación de red social para jugadores de juegos de mesa

Autor: Jorge Gómez Conde
Tutor: Miguel Ángel Mora

julio 2019

Jorge Gómez Conde
Aplicación de red social para jugadores de juegos de mesa

Jorge Gómez Conde

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Agradecimientos

A mi tutor Miguel Ángel Mora que me ayuda y aconsejado en cada paso dado en este proyecto.

A mi padre y a mi madre, por darme la oportunidad de estudiar y estar a mi lado siempre, pase lo que pase.

A Diego y a Elena, que siempre siempre que he necesitado consejo y despejar mi cabeza de dudas.

A mis amigos, simplemente por aguantarme. Especialmente, a Gorka, Bardon, Jorge, Novillo, Claudio, Timor, Jaime y Dani por prestarse a probar este maravilloso proyecto.

A mis compañeros de grado, por hacer más ameno el camino recorrido juntos.

Y a todos los programadores software que mantienen con vida la comunidad y todas las herramientas que he necesitado.

Resumen

El número de aplicaciones móviles ha ido creciendo en una gran cantidad durante los últimos años. Ofreciendo múltiples servicios, algunas para juego, gestión personal de cualquier tipo de actividad, o redes sociales enfocadas a un sector concreto. Pero esto no quiere decir que todas se desarrollen de la misma forma ni con las mismas herramientas.

El proyecto ShareBoard consiste en desarrollar una aplicación para crear una red social para jugadores de juegos de mesa. Además de el interés en el mundo de los juegos de mesa, se trata de generar una aplicación híbrida, tanto web como móvil, que experimenta en su implementación con tecnologías software.

En el documento se recoge un análisis de la evolución de Android, y que tecnologías y frameworks han permitido una expansión de los modelos de desarrollo de aplicaciones.

De entre las distintas formas de crear una aplicación recogidas, se incluye un proyecto software dividido en sus correspondientes fases. En primer lugar, un análisis que describe cuales son los objetivos y el alcance de ShareBoard, junto a los modelos y los escenarios de uso. En segundo lugar el diseño llevado a cabo, incluyendo que arquitecturas software se han utilizado tanto para la ubicación del software como el planteamiento del mismo, reflejado en un patrón de diseño.

Y posteriormente que decisiones se han tomado para su desarrollo, es decir, se describe la metodología de elegida para el proyecto y en que fases es dividida. A continuación, las pruebas realizadas para evaluar el estado del proyecto y su acogida en un entorno real.

Palabras clave

red social, aplicación, híbrida, arquitectura, modelo, patrón, diseño, metodología

Abstract

The number of mobile applications has been growing a lot during the last years. Offering multiple services, some for gaming, other for personal management of any type of activity, or social networks focused on a specific sector. But this does not mean that they all develop in the same way or with the same tools.

The ShareBoard project consists of developing an application to create a social network for board games players. In addition to the interest in the world of board games, it is about generating a hybrid application, both web and mobile, which is experienced in its implementation with software technologies.

The document includes an analysis of the evolution of Android, and that technologies and frameworks have allowed an expansion of application development models.

Among the different ways of creating a collected application, a software project divided into its corresponding phases is included. First, an analysis that describes the objectives and scope of ShareBoard, together with the models and usage scenarios. Second, the design carried out, including that software architectures have been used both for the location of the software and the approach of the same, reflected in a design pattern.

And later that decisions have been made for its development, that is, the chosen methodology for the project is described and in which phases it is divided. Then, the tests carried out to evaluate the status of the project and its reception in a real environment.

Keywords

social network, application, hybrid, architecture, model, pattern, design, methodology

Índice

Listas

Lista de figuras

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Aplicación de red social para jugadores de juegos de mesa

Autor: Jorge Gómez Conde
Tutor: Miguel Ángel Mora

julio 2019

Jorge Gómez Conde
Aplicación de red social para jugadores de juegos de mesa

Jorge Gómez Conde

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Jorge Gómez Conde

Julio 2019

1

Introducción

Durante la último quindenio, con la aparición de los teléfonos inteligentes se han desarrollado un gran número de aplicaciones informáticas que han demostrado de ser de gran utilidad y han sido capaces de conectar a mucha gente que a través de la red, que de ser de otra forma, tal vez, nunca hubieran coincidido.

Este fenómeno es lo que conocemos como redes sociales. A lo largo de los años ha adquirido un papel cada vez más y más importante e influyente en la sociedad moderna. Apareciendo cada día más en mas aspectos de la vida de la personas, pero como todo en exceso, también aportan efectos o condiciones negativas, como por ejemplo la exclusión social o la adicción a los teléfonos móviles [1]

Ahora las nuevas tecnologías se pueden utilizar para unir a la personas que comparten los mismos gustos, creando un tejido social y facilitando la búsqueda de personas por las que se comparten las mismas aficiones.

1.1. Motivación

Es relevante para el desarrollador del proyecto, no solo crear una red social, si no que esta también sea empleada correctamente, minimizando los efectos negativos y aportar cuanto más valores positivos de convivencia. Esto es uno de los principales motivos, pero no el único.

La afición por los juegos de mesa es otra de las razones por las que se ha considerado realizar el proyecto. En combinación con la idea anterior, aparece un proyecto con un problema a resolver. Dado que vivimos en un mundo que progresivamente es conducido a una realidad virtual. A diferencia de los juegos digitales, pueden a portar muchos más valores positivos y beneficios para las personas, tanto como para niños como para adultos. Como son la creatividad, desarrollar habilidades sociales y mejorar la capacidad de inteligencia, sobre todo emocional [2]

También, la inspiración surge de otras aplicaciones parecidas como ”Timpik”[3], una red social para realizar eventos deportivos, como jugar un partido de fútbol o de tenis. Otra de las ideas que se comprendieron para llevar la realización de este proyecto era que habiendo un gran número de

aplicaciones que sirven para la interacción entre personas, no existiera una aplicación de juegos de mesa que estuviera extendida por el mundo digital. Aprovechando este hueco en el mercado, se aúnan tres ideas, los beneficios delos juegos de mesa, reducir la independencia de las redes sociales a través de otro tipo de uso de las mismas y del agujero en el mercado que nos servía este tipo de necesidades aparentemente.

1.2. Objetivos

El objetivo del proyecto es crear una aplicación que sea accesible desde un navegador web o desde una aplicación Android para la participación en la red social. Por esta razón, la aplicación debe permitir el registro, el acceso, la visualización de contenidos, juegos y reuniones y la participación en partidas reales con otras personas.

La aplicación debe facilitar al usuario común la interacción. Es decir, la interfaz debe ser amena sin crear una curva de aprendizaje alta, que permita adaptarse a su uso sin complicaciones, exponiendo los datos de tal forma que resulten visibles y claros.

Desde un punto de vista más técnico, se pretende experimentar la creación de una aplicación tanto web como móvil desde un entorno de programación único, prescindiendo de tener que realizar dos proyectos separados paralelamente. De este modo, también se requerirá un software reutilizable en su apartado gráfico y, por otra parte, robusto y modulado en su parte interna. Este concepto se denomina dentro del mundo del software como *"learn once, write anywhere"*.

Sin embargo, no hay que olvidar que es un proyecto software, y como tal, se plantearan los objetivos propios de comenzar un software desde el principio. El análisis, diseño, pruebas e integración del software son fundamentales para la puesta en marcha de la red social. La mantenibilidad, también será un requisito del proyecto pero no un objetivo final, pues no existirá tiempo para analizar y documentar esta fase.

1.3. Organización de la memoria

La memoria contiene las partes propias del desarrollo de un proyecto software. Como partes adicionales, se incluye un estado del arte y un anexo final que servirá para explicar más profundamente algunas partes del proyecto.

En primer lugar se describe el estado del arte, en el que se ha realizado una investigación sobre la evolución de Android y las tecnologías aplicadas al desarrollo de aplicaciones tanto móviles como de entorno web.

En segundo lugar, se explica el análisis y el posterior diseño del proyecto. Describiendo que requi-

sitos finales se han concluido y que es lo que se espera afrontar en el desarrollo.

En tercer lugar, el desarrollo definirá las decisiones tomadas para implementar todas las decisiones tomadas en el diseño. También incluye aquellos cambios que se han realizado y no se habían previsto en la fase anterior.

En cuarto lugar, el documento recoge las procesos realizados para la integración y posteriormente las pruebas realizadas con usuarios.

Finalmente, la conclusión resultante del proyecto. También, una lista de funcionalidades, requisitos, implementaciones y cambios que habría que desempeñar para la mejora de la aplicación.

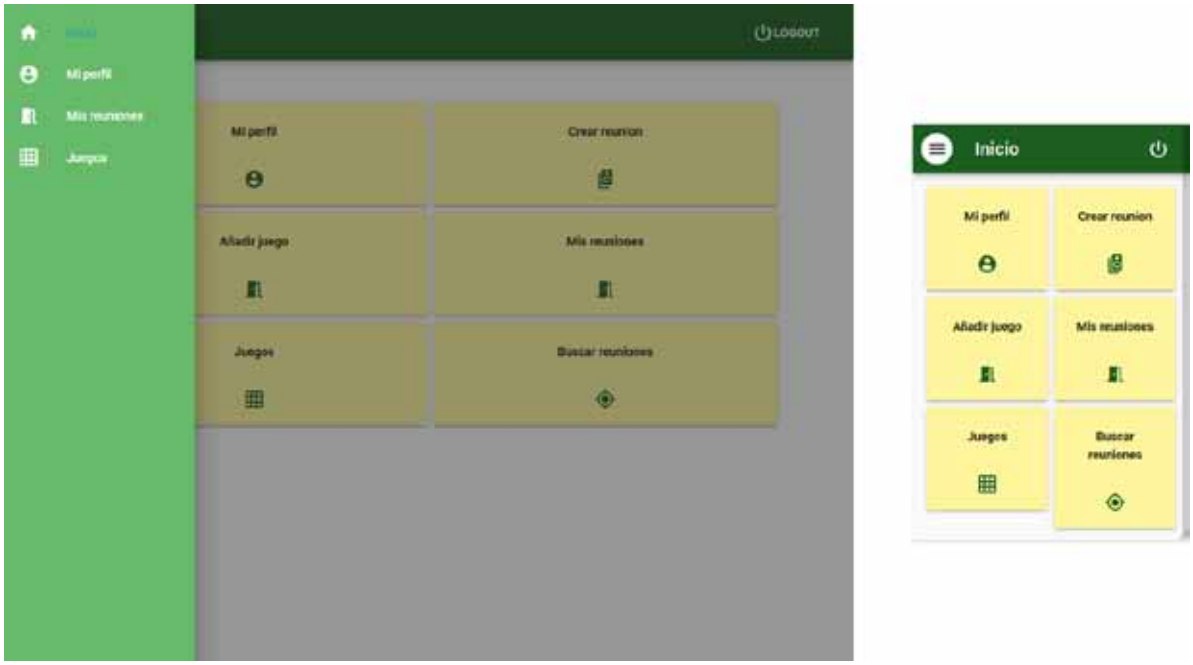


Figura 1.1: Página de inicio de la aplicación en versión web y móvil

2

Estado del arte

Esta sección resume la investigación realizada de las materias relacionadas con las tecnologías aplicadas en el proyecto. En un primer lugar se realiza un análisis sobre la evolución de Android y que características han permitido la expansión del desarrollo de aplicaciones móviles.

A continuación, se analiza las tecnologías híbridas que pueden ser utilizadas para crear tanto aplicaciones web como aplicaciones móviles. Finalmente se añade una conclusión explicando porque se han elegido ciertas tecnologías para la creación del proyecto ShareBoard.

2.1. Primeros pasos de Android

Android fue creado en 2003 por Android Inc, el ingeniero a cargo de la creación fue Andy Rubin. A penas con casi dos años de existencia, fue comprada por Google en 2005, manteniendo el desarrollo en secreto durante dos años más. Es en noviembre del 2008 cuando sale al mercado el HTC Dream/T-Mobile G1, el primer smartphone con Android instalado. Desde este momento, Android no ha dejado de crecer hasta convertirse en el sistema operativo instalado en la mayoría de los teléfonos móviles actuales.[4]

Para ello, Android se ha basado en ser una plataforma de código abierto, con un kernel Linux, que le otorga la garantía de poder ejecutarse en cualquier dispositivo. Otra de las ventajas del uso de código abierto, es permitir a fabricantes, operadores y desarrolladores crear y adaptar funcionalidades para sus propios dispositivos. Con la ventaja de ser multiplataforma y gratuito, permitiendo su extensión a más tipos de terminales, no solo a smartphones.

La potencia de Android reside en su versatilidad, que le sirvió para abrirse hueco en el mercado donde cada marca desarrollaba su propio sistema operativo para sus propios dispositivos, como Blackberry o Apple. Y con ello a su vez, las tecnologías de software que se han ido adhiriendo a Android, han hecho crecer sus funcionalidades acaparando la mayor parte del mercado.[5]

2.2. Evolución de las versiones Android

Al comienzo de Android, en su versión 1.0 lanzada en 2009, tenia unas pocas funcionalidades, que aún perduran, como son la integración de Gmail o Google Play, antes llamada Android Market. No fue hasta la versión 2.1, cuando fue instalada la navegación con Google Maps. Pero, la versión que supuso la expansión del sistema entre el gran público fue la 2.3 Android 2.3 Gingerbread. [6]

Una de las características principales que permitió la expansión del desarrollo de aplicaciones móviles fue WebView y su actualización en Android 4.4 KitKat, que implementaba el motor V8 de Chrome, logrando que fuera capaz de interpretar JavaScript y los nuevos estándares Web. Pero aun mantenía un problema de seguridad, al estar integrado en el sistema no podía ser actualizado con facilidad. Así que hasta que llego Android 5.0 Lollipop, WebView no fue integrada como APK (Android Application Package)[7] para que recibiera las actualizaciones independientes del sistema. Junto a esto, y la necesidad de crear aplicaciones rápidamente, comenzó abrirse el mercado, y con ello la de desarrollar más ágilmente. Fue en 2009 cuando la empresa Nitobi lanzó Phonegap, y con el tiempo su versión de código abierto cedida a Apache, recibiendo el nombre de Apache Cordova, framework para desarrollo de aplicaciones móviles basadas en lenguajes utilizados para la paginas web, HTML, CSS y JavaScript. La ventaja es que permitía desarrollar aplicaciones móviles pequeñas y medianas, básicamente cuando la mayor parte del procesamiento fuera en el servidor, con un menor numero de programadores y simplificando la codificación.

Figura 2.1: Logo de Android 9.0 Pie.

Así comenzó a abrirse el camino para las aplicaciones híbridas frente al desarrollo nativo.

2.3. Los entornos multiplataforma y los entornos nativos

Hay muchos aspectos a tener en cuenta a la hora de elegir el entorno sobre en el que programar una aplicación para móviles. El potencial de integración con otros sistemas, el uso de los sensores, el consumo de batería, la complejidad de testeo de la aplicación o la creación de un proyecto incremental.

Pero las diferencias claves entre ambos desarrollos son tan solo dos. El coste de desarrollo y el rendimiento. Tal vez, si se quiere desarrollar juegos o aplicaciones de procesamiento de imágenes sea mejor una aplicación nativa, pero si se desea implementar una aplicación comercial sencilla, las

diferencias de rendimiento serán pequeñas o inapreciables. Y resulta mucho mas obvio, que un proyecto tendrá un menor coste de desarrollo si escribiendo un código es valido para distintos dispositivos que tener que desarrollar un código para cada dispositivo. Estas ventajas plantean que observemos la dificultad de crear una interfaz.[8]

Aunque cada desarrollador pueda controlar muchos aspectos de la experiencia de usuario, deben seguir unas guías de estilo para mantener una interfaz común con todo el sistema, que son implementadas por un buen abanico de SDKs (Software Development Kit). De esta forma, permite a los desarrolladores elegir una opción que se ajuste más a sus necesidades.[9].

Como una de las necesidades más importantes para este proyecto es la creación de una interfaz gráfica y es el desarrollo ágil, y conociendo que existen una muchos entornos de desarrollo, nos centraremos en los entornos multiplataformas o híbridos.

2.3.1. Tecnologías híbridas para el desarrollo multiplataforma.

Las tecnologías híbridas se basan en una idea muy sencilla, *"learn once, write anywhere"*. Quiere decir que el programador se basta con conocer un solo lenguaje para desarrollar en varias plataformas, con sus lenguajes independientes. Para Android es necesario conocer Java y las librerías orientadas a esta plataforma, lo mismo sucede para IOS con Objective C++... Estas tecnologías han venido a facilitar y agilizar los procesos de creación de aplicaciones multiplataformas. Trabajan en combinación con otros frameworks de desarrollo de aplicaciones web, mayoritariamente basadas en JavaScript.

Anteriormente se ha hablado de Apache Cordova, la versión de código abierto de PhoneGap, para la transformación de aplicaciones web en móviles. No son las únicas plataformas existentes para el desarrollo, también están Xamarin, Ionic y React Native entre las mas conocidas. Las iremos describiendo en orden cronológico.

Ionic apareció en 2013 como proyecto para llevar el desarrollo web a aplicaciones móviles. Basado en AngularJS en sus comienzos, su uso a decaído con los años precisamente también el des huso de Angular, y su curva de aprendizaje mas alta [10]. Aunque ahora admite otros frameworks (React o Vue), su funcionamiento ha terminado basándose en Cordova, permitiendo incluso tomar plugins propios de este framework. No es la API más extendida, ni con mejor fama, aunque es una buena opción para el desarrollo de aplicaciones.[11].

Facebook entró a la competencia en este terreno con React Native. En un principio, y aun sigue estando entre los principales frameworks de este campo, React nació para el desarrollo web, y que con el tiempo, los desarrolladores de Facebook decidieron crear el proyecto React Native en 2013, y publicado oficialmente en enero del 2015, para facilitar la implementación aplicaciones híbridas. Ofrece, como ya se ha dicho, un desarrollo basado den React y JavaScript, con Hot Reloading (permite visualizar los cambios de interfaz sin recompilar el proyecto entero) y que permite el uso de código

nativo dentro de su propio código pero solo se limita a Android e IOS. [12]

Desde sus comienzos, Xamarin, ha sido un framework de desarrollo en .NET. Da posibilidad de crear aplicaciones para Android, IOS y Windows Phone. No es de los entornos mas expandidos, pero al ser el único que utiliza este lenguaje, le permite encontrar hueco en este mercado apoyándose en los desarrolladores que les gusta este lenguaje. [13]

2.4. Conclusión

Al comienzo de Android, las aplicaciones híbridos eran demasiado inseguras y las tecnologías basadas en entorno web para móviles estaban orientadas para pequeñas funcionalidades, conectarse a una página puntualmente que proveía servicios de algún tipo, autenticación o pago, por ejemplo.

El desarrollo de aplicaciones híbridas a ido poco a poco ganando terreno, también dada por la necesidad de empresas y desarrolladores de crear productos aplicados a las nuevas tecnologías más rápidamente. No por ser mejores que las aplicaciones nativas, si no por que su forma de desarrollo esta orientada a metodologías más ágiles y a su menor coste de producción. Algunas mejores que otras, ya que no todas se apoyan en los mismos lenguajes, tampoco dan soporte a todas las plataformas y no tienen todas las mismas opciones de desarrollo. Finalmente, se ha decidió realizar la implementación del proyecto ShareBoard en Apache Cordova al ser la más extendida y que más plataformas abarca. El uso más prologado de esta API por otros desarrolladores, también servirá de apoyo a la hora de resolver problemas y obtener plugins.

Figura 2.2: Logo de Apache Cordova

Otra de las razones por las que se ha decidido a utilizar Cordova, es los frameworks de desarrollo con los que es compatible, ya que la implementación se realizara sobre Nuxt, un framework para el desarrollo Web basado en Node.js que abstrae el desarrollo del paradigma de cliente/servidor basado en el framework Vue de JavaScript.

3

Diseño

3.1. Análisis del proyecto

3.1.1. Introducción

Propósito del sistema

El propósito del proyecto es el de crear una red social o comunidad de jugadores de juegos de mesa. Su función principal será como herramienta de organización de reuniones entre usuarios de la aplicación. Su funcionalidad no es mantener juegos y partidas online y ni tampoco su función principal no será mantener canales de conversación para los usuarios, es decir, un chat. Por otra parte, la aplicación si servirá como base de datos de juegos de mesa, para que los usuarios puedan estar al corriente de sus juegos favoritos.

El fin es que sea una herramienta útil para las personas que disfruten los juegos de mesa y se encuentren en lugares nuevos, o en su localidad de siempre, donde poder encontrar a otras personas interesadas en los juegos de mesa, y formar un tejido social que haga crecer, acerque, y de acceso los juegos de mesa a un número mayor de personas

Ámbito del sistema

Al ser una red social o comunidad, lo usuarios deben estar registrados. Por lo tanto para utilizarla es necesario permitir un sistema de registro en el sistema ”Sing up” y de entrada "Log in” a la aplicación.

Una vez registrados los usuarios podrán cambiar sus características que son mostradas en su perfil y que son visibles para el resto de usuarios. También cada usuario podrá marcar a otro usuario como su amigo.

Las reuniones contendrán usuarios y juegos, además podrán ser privadas y estar abiertas al resto de usuarios o no. Se permitirá la creación ilimitada de reuniones y de juegos, ya que se da por hecho en la buena fe de los usuarios de no crear reuniones y juegos innecesariamente. Cada reunión dispondrá de una fecha, ubicación y un número máximo y mínimo de asistentes, ya que son dos datos muy

relevantes a la hora de reuniones.

El creador tendrá permitido añadir y borrar usuarios de la reunión. Cada usuario es libre de entrar y salir de una reunión mientras se encuentre abierta o no supere el número máximo de jugadores.

Los usuarios podrán añadir juegos libremente a la base de datos. Solamente el creador del juego podrá cambiar los parámetros del juego. Los juegos se podrán añadir a las reuniones para que todos los jugadores sepan a que se disponen a jugar. Además, cada jugador puede marcar, o desmarcar, un juego como favorito.

Otro elemento a disposición de los usuarios son los grupos. Estos servirán para coordinar reuniones entre usuarios con una afinidad distinta a la de amigos. Esta afinidad, puede ser por frecuentar los mismos espacios o por jugar a una misma categoría o tipo de juegos. Desde los grupos, públicos o privados, se pretende facilitar la conexión entre usuarios habituados a unos mismos juegos en un mismo lugar. El acceso del mismo pertenecerá al usuario creador del grupo.

Además de los grupos, también se incluyen los espacios, que son lugares disponibles para la realización de reuniones. Si un usuario posee o conoce una ubicación para la realización de eventos, puede que quiera compartirlo con el resto de la comunidad o con un grupo.

También se tendría en cuenta acciones propias de las aplicaciones de las redes sociales como las notificaciones. El sistema debe tratar de mantener al usuario informado con la datos mas recientes, o bien, con aquella información que mas pudiera interesar al usuario, permitiéndole elegir que considera importante.

Objetivos y criterios de éxito del proyecto

A continuación se declaran los principales objetivos de la aplicación para considerar que se ha alcanzado el estado mínimo éxito. Respecto al nivel de acceso, debe ser posible entrar desde un dispositivo Android o desde un navegador web Mozilla Firefox, Google Chrome o Safari. A nivel practico de usuario, es necesario un mínimo de requisitos:

1. Crear reuniones, con un mínimo de características, que son fecha, lugar, juegos y ubicación.
2. Gestión de reunión de las características secundarias y de la participación de los usuarios. Es decir, que puedan se puedan añadir o borrar jugadores de la reunión.
3. Crear juegos, con las características mínimas de numero de jugadores, dificultad y duración.
4. Gestión de juegos, cambiar características mínimas.
5. Gestión del perfil de usuario.

6. Unirse o salir de reuniones como usuario normal.

7. Notificaciones de avisos y cambios a tiempo real.

Pero también hay requisitos no necesarios pero muy útiles para la inmersión y experiencia de uso del sistema:

1. Evaluar reuniones.

2. Enviar invitaciones a una reunión.

3. Uso de grupos de usuarios para crear reuniones.

4. Uso espacios como ubicación para reunión.

Más adelante, en el apartado de los Casos de Uso se describen de mejor forma las funcionalidades que pretenden lograr y que serán tomadas como éxito de la aplicación.

Definiciones, acrónimos y abreviaturas

Usuario Personal normal como usuario del sistema. Definida con email, nombre y foto. Como tal tiene capacidad para crear cualquier elemento en la aplicación

Administrador o creador Un administrador es un usuario dueño de una reunión o dueño de un juego. Ser dueño significa haber creado el juego o reunión y tener la capacidad de gestionarlo, como por ejemplo, cambiar el nombre de la ubicación de la reunión.

Juego Representación virtual de un juego de mesa. Mantendrá unas características mínimas para mostrar en su propio perfil.

Reunión Es la representación virtual del espacio y lugar donde tendrán que reunirse los usuarios participantes. También mantiene información sobre que juegos se jugarán y durante cuanto tiempo se prevé que dure la reunión.

Formulario o evaluación Cada reunión tiene un formulario de evaluación sobre como ha transcurrido la reunión, apuntando los ganadores y/o algún incidente que hubiera surgido con uno o varios usuarios.

Grupo de usuarios Es el un conjunto de usuarios que ha decidido unirse para formar un grupo. Puede servir para ayudar a nuevos usuarios a integrarse en una comunidad cercana, por ejemplo.

Espacio Pueden definirse como locales o ubicaciones privadas o publicas creadas por los usuarios. Un espacio privado puede ser un local de un comercio que cede su espacio para los jugadores o zonas publicas como mesa de un parque o espacios disponibles en un centro social.

3.1.2. Descripción del sistema

La descripción del sistema contara con escenarios y casos de uso. El objetivo de los escenarios es definir una funcionalidad concreta y esperada por el usuario durante el uso del proyecto. Por otra parte, los modelados de los caso de uso nos permite ver una interacción mayor por parte del cliente con el proyecto.

Escenarios

En esta sección se han escogido los escenarios que mejor describen el uso del sistema. Por otra parte, se han omitido algunos casos por mantener similitudes de funcionamiento escenarios ya incluidos. En la descripción de cada escenario, se puede describir una alternativa, o problema, que abre un nuevo camino de desarrollo. A su vez puede suceder que se involucren mas elementos del sistema, esto también se reflejará en la descripción de la alternativa o problema. Cada escenario puede tener sus propios problemas, o no tenerlos, al igual que puede que no haya surgido alguna alternativa.

Para ver la descripción de los escenarios de uso consulta el anexo.

Modelado de casos de uso

La modelización de los casos de uso se ha llevado a cabo en dos diagramas diferentes porque existen acciones exclusivas para un tipo de actor. Hay que aclarar que ambos actores están al alcance de cualquier usuario registrado, pero el hecho de crear elementos en el sistema dará lugar a mas casos de uso, por eso se ha representado con dos actores, un usuario normal y otro usuario administrador.

Ver anexo para ver los escenarios modelizados.

Casos de uso

En este caso, el diagrama de casos de uso se distinguen dos tipos de claros de actores, un usuario corriente y un usuario que es creador de los elementos de la aplicación, juegos, reuniones, espacios y grupos. Las acciones de creación están al alcance de cualquier usuario, pero son precisamente estas acciones las que distinguen que un usuario administrador, creador de un elemento, de un usuario normal. Los administradores, como se explica en el subsección anterior, son aquellos que han creado un elemento en sistema.

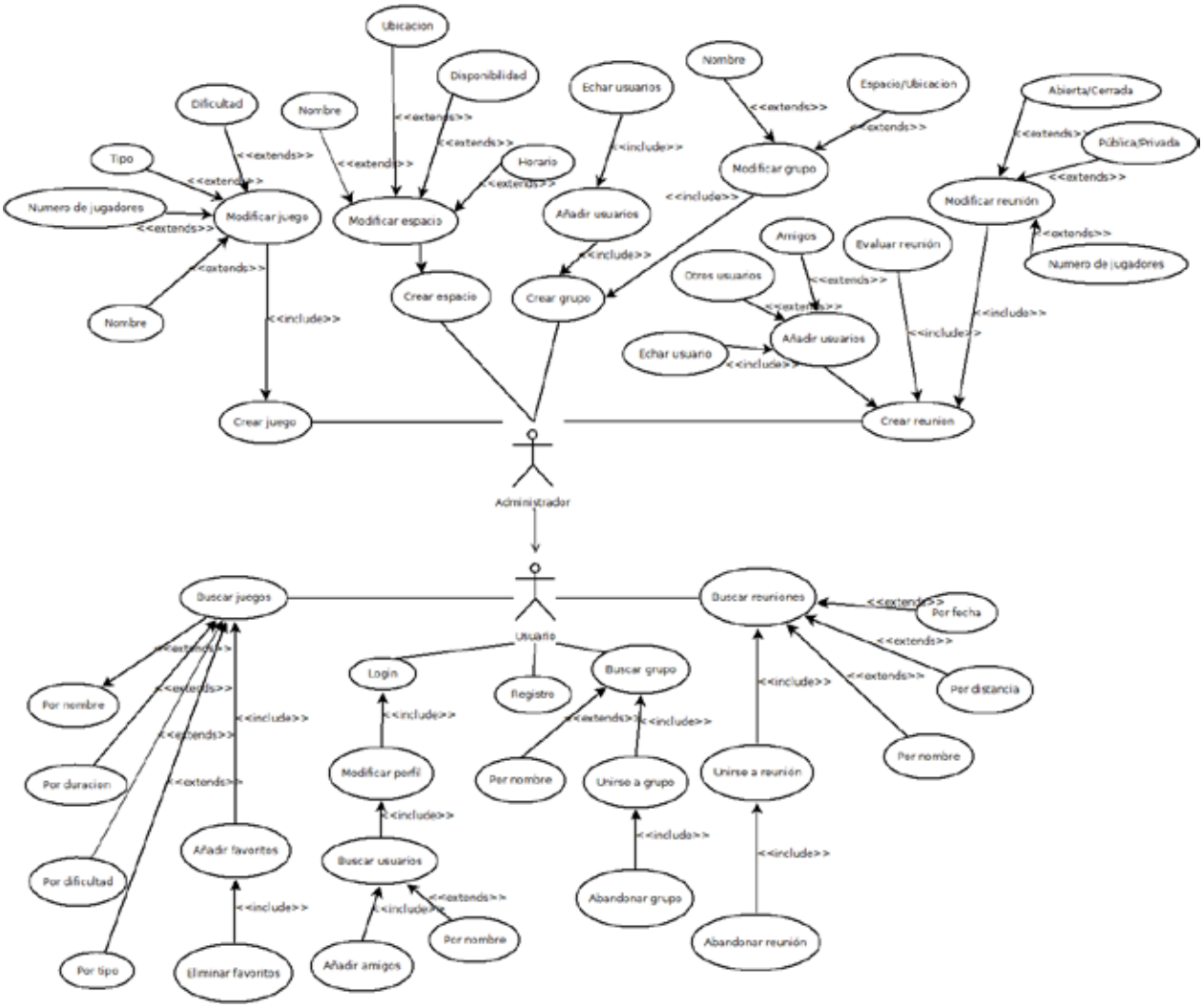


Figura 3.1: Diagrama de casos de uso

3.1.3. Requisitos del sistema

El análisis anterior ha servido para recoger los requisitos que definirán la aplicación. Se dividen en dos categorías, siendo los requisitos funcionales aquellos recogidos para satisfacer lo que debe hacer la aplicación respecto al usuario, y los requisitos no funcionales, que se orientan a como debe funcionar y responder la aplicación.

Requisitos funcionales

Los requisitos funcionales se agrupado de la siguiente forma. En un primer lugar se describen los requisitos que se han considerados los mas importantes y comunes para todos los usuarios, antes de que el usuario haya creado un elemento. Continua con las interacciones que tiene el usuario con el resto de elementos, es decir con otros usuarios, juegos, reuniones, grupos, espacios y evaluaciones.

Los requisitos funcionales están enumerados en el anexo.

Requisitos no funcionales

Los requisitos no funcionales están orientados a propiedades del sistema que sirven para especificar el sistema como un todo. Quiere decir que se definen capacidades o características que son implementadas en el sistema de forma general. En este caso, se describen necesidades de mantenimiento, orientadas a como de debe implementar el software para cuidar su vida o su capacidad de operación para la respuesta del sistema a eventos. También se han tenido en cuenta la seguridad, o especificaciones de la interfaz de usuarios.

Al igual que los requisitos funcionales, estos también se encuentran en el anexo.

3.2. Diseño del proyecto

3.2.1. Introducción

En esta parte del documento se mostrará una modelización del diseño de la red social ShareBoard. No solo incluye modelos y diagramas, también describe como se modelizarán los datos para tratar de conseguir un software modulable y mantenible. Es importante definir bien para la arquitectura del proyecto, ya que servirá de estructura global, y realizar cambios dentro de esta estructura pueden suponer que sean laboriosos y difíciles de implementar. La realización de esta sección ayudará a focalizar la satisfacción de los requisitos y como documento base durante todo el desarrollo. El diseño llevado aquí, se tendrá en cuenta durante todo el proceso software y también servirá como definición del producto final esperado.

En la sección de arquitectura se describirá el modelo que se usarán para llevar acabo el desarrollo de la aplicación, que cubrirá los requisitos funcionales y no funcionales. Esta parte servirá de orientación a los programadores de como entender la arquitectura del software esperado con una modelización de la realidad que enfoca el planteamiento del proyecto.

Las secciones de diagramas se mostrará el funcionamiento esperado de las funcionalidades de la aplicación descritas en la sección de arquitectura. Esta apartado, se centra en una parte visual del proyecto de software, que facilita la compresión de la arquitectura y sus componentes.

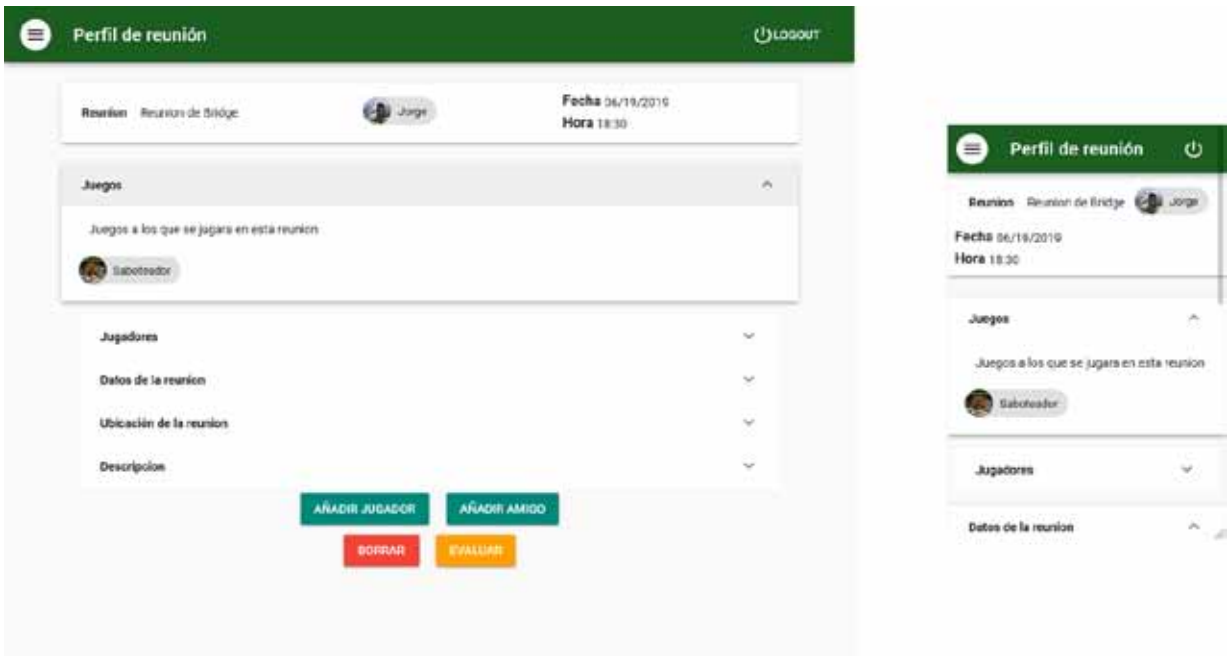


Figura 3.3: Visualización del perfil de una reunión en versión web y móvil

3.2.2. Descripción del sistema

Como ya se ha dicho anteriormente, es importante describir y diseñar una arquitectura del sistema íntegra y que no sea necesario cambiar durante la vida del proyecto. Aun así, es un proceso creativo sustento a fallos, por lo tanto también se debe facilitar la modularidad y la independencia entre las partes del proyecto. Además, que una buena descripción detallada facilitará la búsqueda de tecnologías que se adapten mejor a los requerimientos, funcionales como no funcionales. Dependiendo de esto, podremos potenciar los puntos más fuertes de la aplicación y reforzar aquellos más débiles.

De todas formas, el diseño arquitectónico del sistema se llevará a partir de dos tipos de vistas distintas.

La primera, como se debe implementar la infraestructura del sistema, es decir, donde y como se alojará el software, y para ello se elegido una arquitectura backendless. El motivo es que para garantizar la eficiencia en el desarrollo es conveniente separar lo más posible la implementación de los requisitos funcionales como los no funcionales. De esta forma es el proveedor es quien garantizar el servicio y acceso a una base de datos que mantenga los requisitos no funcionales operacionales y de seguridad.

Y la segunda, como debe formarse y componerse la estructura software. Dado que esta parte es mas abstracta, ya que es la que va a componer la lógica interna de la aplicación y más cercana al usuario, afectara a los requisitos funcionales de la aplicación. Para describirla más fácilmente y obtener modularidad, se necesita diferenciar las partes de lógica, los datos y la interfaz de usuario. En este caso, se ha elegido el modelo de desarrollo MVVM (Modelo Vista Vista-Modelo).

Arquitectura Backendless

Este modelo de arquitectura es muy parecido a cliente servidor en el aspecto en el que se diferencian claramente dos tipos desarrollo, el de una parte orientada al cliente, donde esta el peso de crear una interfaz de usuario, y la de un servidor que gestiona los datos. Solo que se prescinde de implementar un servidor y se toman de un proveedor externo.

La mayor parte del desarrollo de la aplicación esta orientada a su interfaz, por que tiene que servir la mayoría de requisitos que influyen con la interacción con el usuario. La implementación sera desarrollada con Vue y deben estar los recursos disponibles tanto como para Android como para la versión web. Dado que Nuxt permite la creación de aplicaciones SPA (Single Page Applicaction) podemos prescindir totalmente del desarrollo de un servidor propio, y llevar los servicio necesarios fuera.

El proveedor externo debe proporcionar el alojamiento de una base de datos que permita el uso de triggers (disparadores) para las notificaciones, un servicio de autenticación, un servicio de alojamiento web, además de un rendimiento positivo para que la respuesta de la aplicación no influya considerablemente en la experiencia del usuario. Estos servicios son proporcionados por Firebase, sus productos

de Database, Functions, Database y Authentication.

Arquitectura MVVM: Modelo Vista Vista-Modelo

Esta arquitectura, como Modelo Vista Controlador, trata de diferenciar las partes del software que componen interfaz, la lógica y el modelo de datos. Orientada al diseño de interfaces, tanto web como móviles, su objetivo es separar el máximo posible la lógica de la vista, y que sea el modelo quien controle el mayor número de operaciones lógicas posibles.

La vista se encarga de la presentación de los datos e informa a la Vista-Modelo sobre las acciones del usuario. A diferencia del patrón de diseño MVC, la interfaz puede tener eventos y enlaces a datos del modelo, de tal forma que siempre se mantenga sincronizada con la vista-modelo. En este aspecto, hay mas libertad para la vista a la hora de mostrar los datos, ya que no es el controlador quien se encarga de formatearlos, si no la propia vista es quien se ocupa de transformar el dato para su visualización.

En un patrón MVC, es el controlador encargado de enlazar la vista con el modelo, la vista-modelo funcionara de una forma parecida. No solo mantiene un enlace entre ambas partes, también mantiene el estado de las datos y del otras aspectos de la aplicación, como por ejemplo, si la aplicación esta cargando algún dato de la nube.

El Modelo, junto a la vista-modelo, cooperan para la gestión de los datos. Se encarga de mantener la representación real de los datos y del acceso a los mismos.

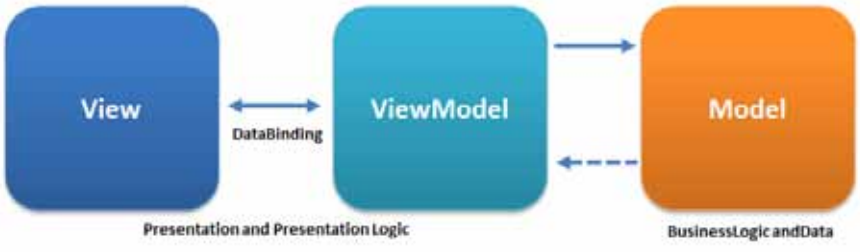


Figura 3.5: Diagrama de las relaciones en una arquitectura MVVM

Vista

En la interfaz se implementará todo aquello que el usuario podrá visualizar y sobre lo que interactuará. La vista interactúa con las acciones realizadas por el usuario, creando un flujo de eventos entre ella y la vista-modelo, pero de una sola dirección, de tal forma que la vista tiene referencias a la vista modelo, pero no al revés.

VistaModelo

En esta arquitectura la Vista-Modelo funciona de una forma parecida al controlador en un modelo MVC, solo que en lugar de que el controlador sea independiente a la vista, en este caso, la vista-modelo se comporta como un contenedor del estado de la aplicación y de la lógica. Para explicar esta parte se entenderá solamente como si fuera el controlador, ya que esta parte es donde se implementarán sus métodos y donde se almacenará los datos visualizados por el usuario.

Modelo

Esta sección describe como se implementara la parte servidor del sistema y parte tendrá que ser implementada en el frontend. Es decir, proporcionar la información y la que se encargara de guardar, tanto fuera como dentro de la aplicación en ejecución. En parte funciona como modelo del patrón MVC, dentro de la aplicación mantiene los estados y definición de los datos mostrados y a su vez trabaja con el proveedor para la integridad de la información.Esta parte también debe proporcionar de algún modo para servir notificaciones al usuario cuando este no este utilizando la aplicación en una plataforma Android.

En la sección de diagramas del anexo se encuentra el diagrama de clases que sirve de descripción del modelo. Con el diagrama, se puede ver la relación que hay entre los elementos de la aplicación.
A continuación se describen que representan y que se espera de ellos.

- **Usuario:** Contiene toda la información relativa al usuario, el correo electrónico, el nombre, la imagen de perfil, la descripción personal y las notificaciones en las que esta apuntado. Respecto a otros usuarios, guarda una lista de amigos con los identificadores de otros usuarios.En relación a otras clases guarda su lista de juegos favoritos. Los métodos contenidos aquí servirán para la gestión del perfil del usuario y añadir y eliminar amigos de su lista. También incluirá filtros de búsqueda de usuarios. No necesitamos guardar la contraseña de usuario en la base de datos, ya que es mantenida por Firebase Authentication.

- **Juego:** contiene la información relativa de los juegos. En los juegos se guarda el nombre, la imagen del juego, el tipo, la dificultad y el tiempo aproximado de duración de la partida. No contiene métodos complejos, se limitan a filtros de búsqueda, constructor de la clase, cambios y obtención de las propiedades de la clase.

- **Reunión:** contiene la información relativa a las reuniones, que es nombre, la fecha, la hora, la ubicación, la descripción y el numero máximo y mínimo de jugadores. En relación con otras clases, guarda el identificador del usuario creador de la reunión, una lista de usuarios que están dentro de la reunión y una lista de juegos.

- **Grupo:** contiene información a los grupos, que es el nombre, la imagen, la ubicación habitual de reunión del grupo y una descripción. Respecto a otras clases, incluye una lista de usuarios

pertenecientes al grupo y el identificador del usuario creador. Los métodos incluidos en la clase serian aquellos que sirven para la modificación de datos del grupo, añadir o eliminar usuarios y crear reuniones con valores preestablecidos, como la ubicación de reunión.

- **Espacio:** contiene la información de un espacio, que es el nombre, el correo electrónico, una imagen y la ubicación del espacio. Los métodos serian aquellos que sirven para la modificación y obtención de los valores del espacio.
- **Notificación:** contiene los datos que se mostrarían en una notificación. Como clase es muy sencilla, contendría un titulo y un cuerpo, además de un contenedor para guardar el identificador de algún dato relativo a la notificación, como una reunión, usuario, o juego. Las notificaciones son estáticas, por lo tanto no es necesario crear métodos para la modificación de sus propiedades.
- **Tablón:** contiene y mantiene las notificaciones. La clase tendría dos relaciones con otras dos clases del sistema. La primera estaría relacionada con el usuario al que corresponde el tablón. La segunda es una lista de notificaciones que el usuario titular gestiona. Los métodos deben permitir borrar notificaciones al usuario y añadirlas al tablón por el sistema.
- **Evaluación:** contiene la información resultante de la evaluación de una reunión. Esta clase en particular tiene que recoger información de la clase Reuniones, Juegos y Usuarios. De la clase Reuniones, guardará el identificador de la reunión evaluada. De los usuarios, guarda una lista de ganadores de la reunión, una lista de la puntuación de cada jugador y una lista de incidencias relacionada con cada usuario, si las hubiera. Hay que añadir que cada evaluación contiene una descripción. Los métodos en esta clase se mantienen en la obtención de los atributos de la clase, ya que una vez creada la evaluación no se podrán hacer cambios.

3.2.3. Diagramas

Maquetas de interfaz

Las maquetas describen gráficamente como se espera que sea la interfaz de usuario. Como ya se dice anteriormente en el documento, para la descripción de una ventana puede haber dos modelos distintos que corresponden a la versión Web y a la versión Android.

Como requisitos fundamentales para el uso de una interfaz es que esta sea sencilla y fácil de usar. Esto implica que sea intuitiva y que a un usuario novato no tiene que pasar por una curva de aprendizaje alta para llegar a usar todas las funcionalidades dentro de la aplicación.

Dado que buscamos compatibilidad tanto en entorno Web como en aplicaciones Android, hay que tener en cuenta que la disposición de los elementos no puede ser la misma por la diferencia del tamaño de pantalla. Por otro lado, para reducir la carga de trabajo, se implementara la interfaz buscando la mayor compatibilidad entre la versión Android y Web.

Como característica común, las paginas contendrán un marco en el que se dispondrán los elementos de navegación y un titulo de la pagina en la que se encuentra el usuario en el momento. Es importante destacar, que se tratará de que las páginas se modúlables. Eso significa que la página, o los elementos que la forman sirvan para representar distintos objetos, como por ejemplo, que los elementos del perfil de reunión sean compatibles también para los de un usuario.

En la sección de diagramas, se incluyen maquetas que muestran como se distribuirá los datos en la interfaz.

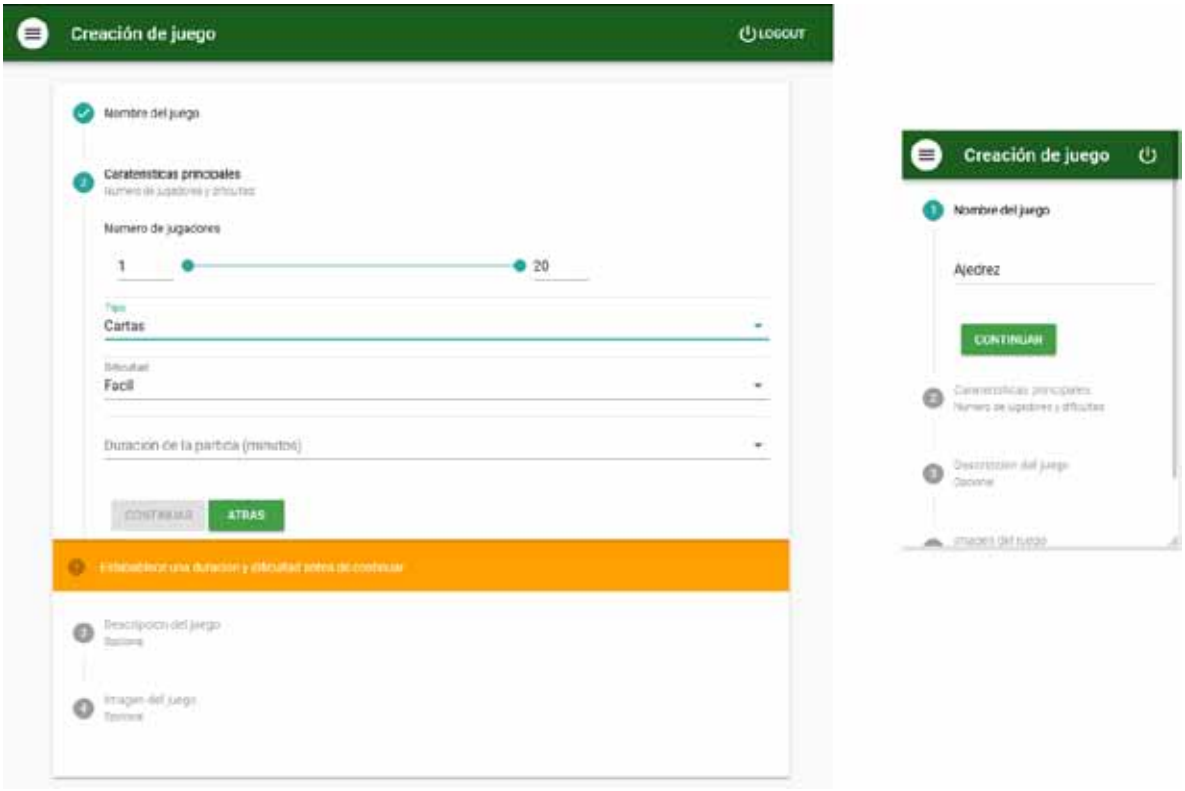


Figura 3.7: Visualización de la creación de un juego en la versión web y móvil

Diagrama de clases

El diagrama de clases describe la relación entre los objetos del sistema. Corresponde a la descripción del modelo del apartado anterior.

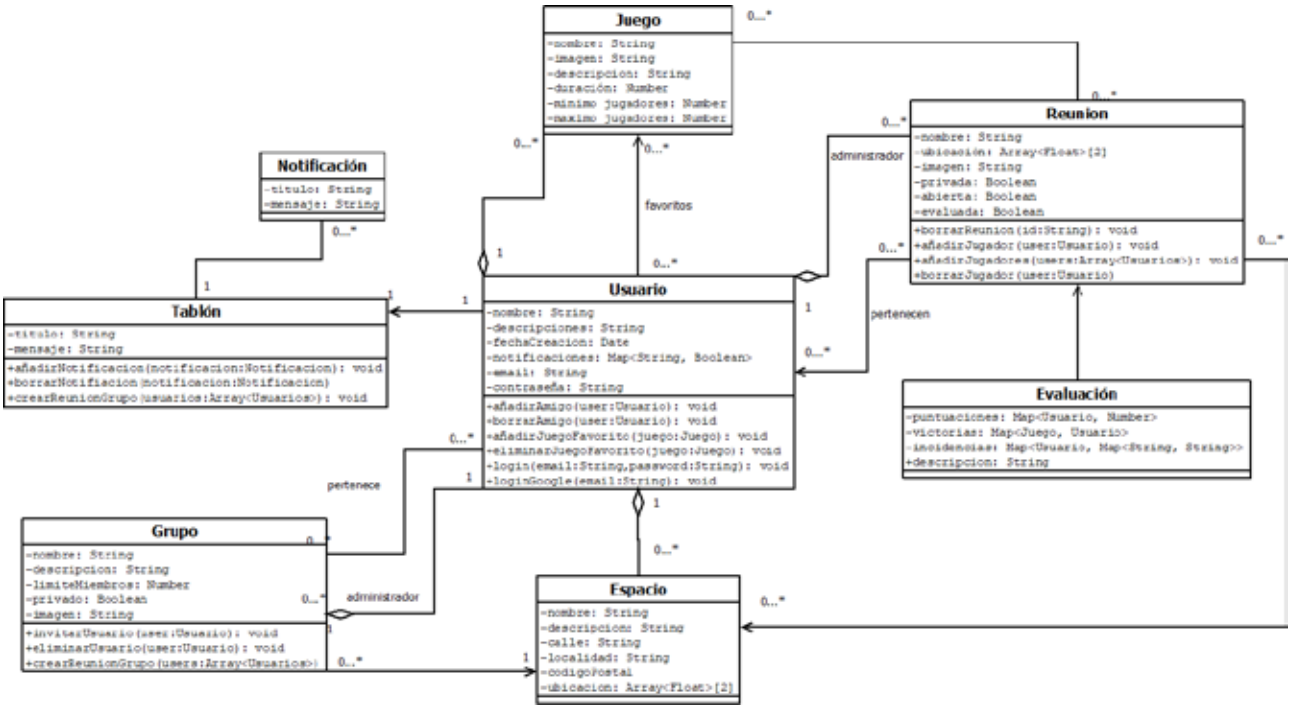


Figura 3.9: Diagrama de clases de la aplicación ShareBoard

3.3. Tecnologías para la implementación

Una vez licitados los requisitos y funcionalidades exigidas por el proyecto, es necesario plantear tecnologías que permitan que funcione el sistema en conjunto de forma eficiente para la mejor experiencia de uso, desde el punto del usuario. Pero también es importante tener en cuenta para el desarrollador, que tecnologías permiten lo anterior, además de facilitar una modularidad, mantenibilidad y robustez en el código.

Para poder explicar esto, hay que observar las conclusiones que se han llegado en el diseño. La infraestructura del proyecto sería backendless, lo que define una parte clara a implementar, y por donde se debe comenzar a plantear de que tecnología es más conveniente para el proyecto. Desde este punto de vista, queda claro que hay dos partes diferenciadas, un parte cliente que será donde se centre el desarrollo y un servicio de servidor dado por un proveedor.

Ahora es el momento de tener en cuenta las clases, u objetos, que se conformarán en su conjunto la aplicación. A si que se escogerán aquellas tecnologías que faciliten la implementación de un aplicación en su parte cliente, que permita el desarrollo con una interfaz sencilla, amena y bonita con un manejo de la visualización cuanto menos complejo y que a su vez permita un desarrollo ágil. Pero para comenzar, se explicará el software, o framework principal del proyecto

3.3.1. Node.js + Nuxt + Vue

Node.js es un entorno de programación orientado a eventos para la ejecución de programas en JavaScript [14]. Se puede considerar una API (Application Programming Interface), que aúna en un proceso común el desarrollo de una interfaz. Una de las mejores ventajas de Node.js es muy fácil comenzar el desarrollo y probar rápidamente el código, lo que otorga una capacidad importante de desarrollo ágil. También permite añadir ampliar los servicios aportados con el gestor de paquetes ”npm” (Node Package Manager), pudiendo añadir módulos de código libre desarrollado por otros programadores.

Para implementar el software base del desarrollo del proyecto, se ha hecho uso de la Nuxt [16]. Nuxt esta construido para el procesamiento de la interfaz abstrayendo al programador del paradigma cliente-servidor principalmente. Servirá como esqueleto del proyecto, organizando los ficheros del desarrollo y simplificando la implementación. La ventaja de Nuxt es que permite crear una aplicación entera, desarrollando también la parte servidor, o bien omitir la creación de un servidor, y crear una SPA, Single Page Application. Para la implementación del proyecto se ha escogido una SPA, que facilita el desarrollo por separado de una interfaz gráfica y complementarla con servicios ya creados para el servicio de una base de datos, tal y como se describe en la arquitectura backendless.

Por otra parte, la principal ventaja de trabajar con Node.js que obtenemos en este entorno de desa-

rrollo es pode trabajar con otros frameworks para ampliar la capacidad de desarrollo. El framework principal es Vue, que esta orientado a la construcción de interfaces de usuario [15]. Vue amplia la potencia del lenguaje de JavaScript, permitiendo extender las etiquetas HTML y de esta forma, poder crear nuevos componentes gráficos. Además otorgar potencia a la hora de manejar estos componentes pudiendo alterar su estado, cambiando los valores que muestran o si deben aparecer o no. Vue cumplirá la función de vista-modelo dentro de la arquitectura, sirviendo como puente y lenguaje común de Vuetify y Vuex, de los que se habla a continuación.

3.3.2. Vuetify + Vue: satisfaciendo la necesidad de la interfaz

El desarrollo de la interfaz ha sido facilitado por Vue y un framework complementario para Vue, Vuetify [17]. Vuetify es un framework de Vue de componentes visuales, que aportan una gran cantidad de elementos visuales y una capacidad de control de la interfaz para distintos dispositivos. La principal ventaja de Vuetify es que permite crear una interfaz, de una forma parecida a HTML, pero permitiendo un mejor control de los componentes y del espacio de la página.

Por lo tanto las necesidades que satisface son las de crear una interfaz única y adaptable a los dispositivos en los que se ejecute la aplicación. Además de que la interfaz debe ser sencilla, la facilidad del uso de los componentes y su visualización ayudará a controlar la entrada de datos a la aplicación, evitando así la introducción de valores o datos erróneos. Todo esto unido a que permite crear componentes estándar que se pueden reutilizar para distintas partes del proyecto.

3.3.3. Satisfaciendo la necesidad del modelo

Vuex

La implementación del control de los datos y su manipulación esta gestionada por Vuex [18]. De nuevo, es utilizado un framework basado en Vue para resolver las necesidades del proyecto. En este caso, Vuex funciona como una librería de patrón de gestión de estados. Esto significa que gestiona los datos internos de la aplicación en un solo sentido y proporciona integridad para el mantenimiento de los mismos para poder transmitirlos al servidor. Es decir los cambios son gestio-

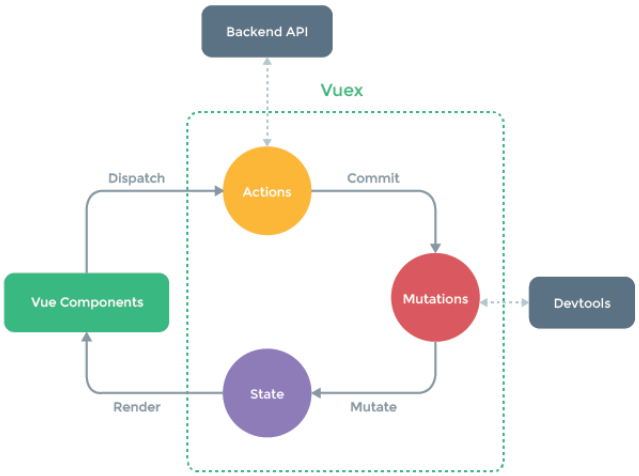


Figura 3.11: Diagrama del flujo del estado de datos en Vuex.

nados por pasos, para que la interfaz
responda y construya correctamente la
visualización de los datos en correspondencia con el modelo.

Para realizar un cambio, la interfaz llama a una acción, o *"action"*, esta a su vez llama a una mutación, o *"mutation"*, que realiza un cambio en los datos estáticos. Los datos estáticos se guardan en un estado, o *"state"*, dentro de Vuex, y pueden considerarse los datos raíz de la interfaz. Las *"actions"* servirán a su vez, para cambiar el modelo de datos almacenado en la aplicación y transmitirlos al servidor proveedor del que se se hablará en el siguiente apartado. El diagrama de la derecha describe el ciclo de cambios de datos dentro de la aplicación detallando el papel de Vuex.

Como hemos explicado, esta infraestructura software sera utilizada para la gestión de los objetos definidos en el diseño sin utilizar variables objetos como en Java o C++.

A pesar de estas diferencias con un lenguaje orientado a objetos normal, aporta modularidad e independencia entre los módulos, así que resultara útil y facilitara el desarrollo, porque se sigue manteniendo al acceso a las clases desde la interfaz en un punto en común.

Servidor de proveedor externo

Como anteriormente se ha dicho, la aplicación será desarrollada y programada omitiendo la parte servidor. Esto quiere decir que se obtendrá servicios externos en lugar de desarrollar desde cero un servidor. Los servicios que se han utilizado han sido los de Google Firebase [19].

Hay varios motivos por los que se ha decido utilizar Firebase en lugar de desarrollar un servidor propio. El principal motivo es el tiempo de desarrollo para ciertos requisitos, como son las notificaciones y cambios a tiempo real. Casi la implementación de un sistema con estas características podría llevarse en un proyecto a parte, porque no solo habría que tener en cuenta el funcionamiento correcto de las funciones, también propiedades de rendimiento o de servicio.

Los servicios empleados de Firebase han sido FireStore, Authentication, Functions, Messaging y Hosting. El primero servirá como base de datos, sus consecuentes funciones de acceso a la misma y además permite el seguimiento a tiempo real de los datos. El segundo, servirá para mantener las cuentas de usuario de los usuarios registrados, dando servicios de cambio de contraseña y registro con otros servicios de autenticación, como es el de Google. En tercer lugar, Functions y Messaging serán útiles para la implementación de las notificaciones. Finalmente, el servicio de Hosting permitirá la implementación y alojamiento de la aplicación en la web.

3.3.4. Portabilidad: Apache Cordova

La última tecnología utilizada, no tanto con el fin de desarrollo, si no de implementar el trabajo realizado en una versión Android es Apache Cordova. Este framework servirá como "traductor"de una aplicación web desarrollada en HTML y JavaScript en una aplicación nativa de Android. Implicará que el desarrollo solo se centre en una parte, que es la aplicación en su versión Web.

Es importante destacar, que las notificaciones nativas de Android han sido específicamente desarrolladas para esta versión. No se ha implementado en la versión web porque normalmente las notificaciones en un navegador no suelen estar activadas por el propio usuario.

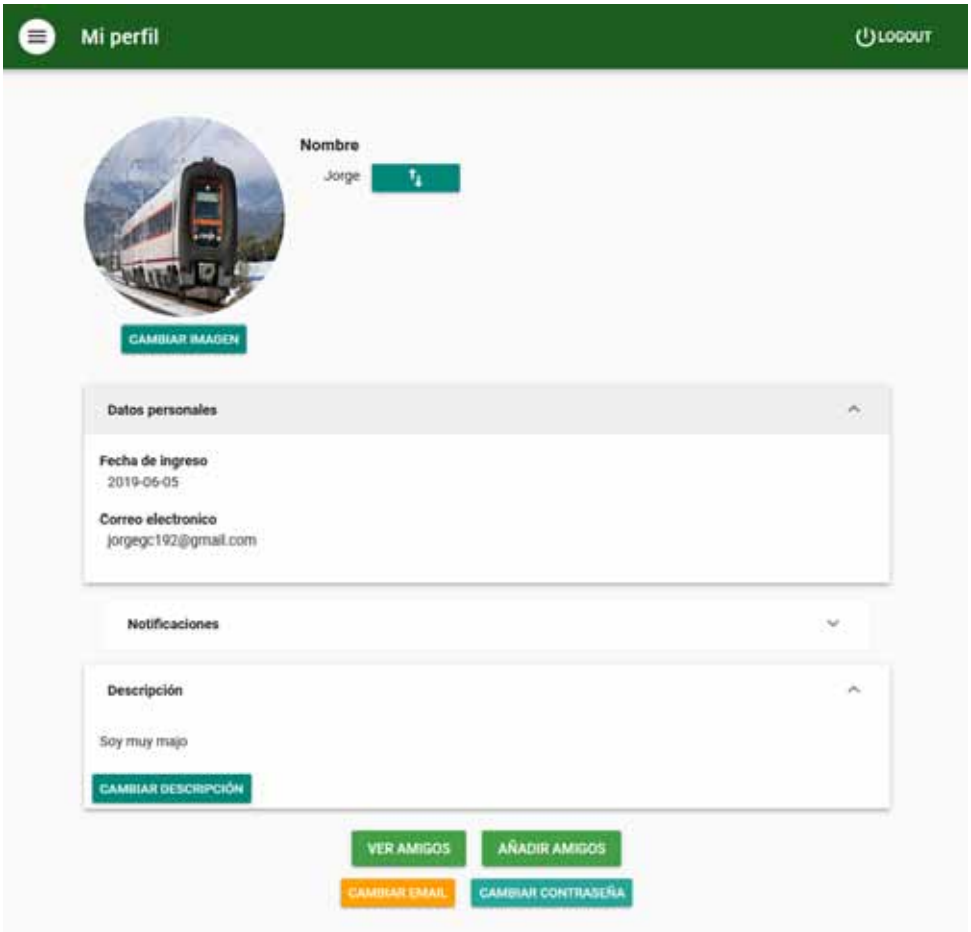


Figura 3.12: Vista de perfil de usuario en la versión web

4

Desarrollo

4.1. Metodología de desarrollo

La metodología a seguir tiene que servir para poder afrontar el proyecto por partes y que poco a poco vaya creciendo. Pero también debe tener en cuenta ciertos requisitos y lineas básicas que no deben cambiar durante todo el desarrollo. Esto implica la combinación de dos estilos de metodologías.

Una parte se enfoca en el progreso de la aplicación de una metodología ágil. El motivo es el tamaño del equipo de desarrollo, que podría dividirse en varios roles, pero concretamente en este proyecto solo existirá un programador, por hay necesidad de comunicar cambios a otros programadores y se puede prescindir algo más de la documentación para la coordinación. También puede que surjan cambios durante el desarrollo, que cambie la implementación de un requisito, y utilizar este tipo metodología facilita afrontar este tipo de cambios. Otro motivo, es el planteamiento de un desarrollo incremental, que ayudará a dividir el proyecto en partes y aislar los problemas o errores que pudieran surgir.

La otra parte se enfoca en desarrollar el proyecto basado en un plan. La razón es que antes de realizar cualquier implementación es exigir cierta especificación con un diseño previo. También se espera que el proyecto cumpla ciertos requisitos mínimos, por eso, se toma como base del proyecto el documento de análisis y el documento de diseño. Ambos documentos marcarán las lineas mínimas que debe seguir el proyecto, siendo estas susceptibles de posibles cambios durante el desarrollo.

Dado que se utiliza este modo de desarrollo híbrido, conviene plantear un progreso iterativo, donde a cada incremento se vayan incluyendo los requisitos planteados en la metodología basada en un plan. Las iteraciones estarán definidas por los requisitos que afronta. A si que es importante, plantear correctamente cuales son las partes mas básicas y esenciales de la aplicación para priorizarlas. Como el proyecto esta orientado a un producto para un usuario, que podría ser cualquier persona, se debe ir aumentando la complejidad de uso poco a poco. Entonces, se comenzará por las partes que servirán como base para el resto del proyecto.

La metodología que mas se ajusta a los requerimientos del proyecto es Scrum. Scrum divide el proyecto en *"sprints"* como unidad de iteración con una longitud temporal. Durante cada sprint se

asignarán requisitos que se afrontan, y al final de cada uno, se revisa si quiere introducir y añadir nuevos requisitos, o prescindir de alguno. Esto ayuda a desglosar el producto en piezas manejables y más comprensibles, y poder visualizar el producto del desarrollo más claramente al final de cada iteración.

Hay que destacar, que después de cada iteración se espera un producto resultante para la parte de desarrollo.No tienen porque ser un producto final y útil para el usuario, pero si tangible y útil, para el programador, y para las siguientes iteraciones. Las iteraciones se componen de los requisitos que acapara del proyecto y el tiempo de desarrollo.

En el anexo, se muestra algunos ejemplos del código de Vue, Vuex y Vuetify.

4.2. Iteraciones

4.2.1. Iteración 1: Funcionalidades básicas

Los requisitos funcionales que se afrontan en esta iteración son:

1.1.1	1.1.6	1.1.15	1.1.20
1.1.2	1.1.8	1.1.16	
1.1.3	1.1.9	1.1.18	

Esta iteración significa el primer contacto con la aplicación. Como parte de la previsión de tiempo, en esta iteración es importante tener en cuenta el tiempo de aprendizaje que conlleva utilizar las tecnologías mencionadas en el punto **"3.3 Tecnologías para la implementación"**.

Para comenzar se implementarán parte de la interfaz gráfica que servirá como base, ya que nos servirá posteriormente para lanzar las funciones que corresponden al modelo. Esto significa que se crearán componentes visuales, como la muestra e introducción de texto, números y valores booleanos, la visualización de los filtros de búsqueda o elementos en listados, que

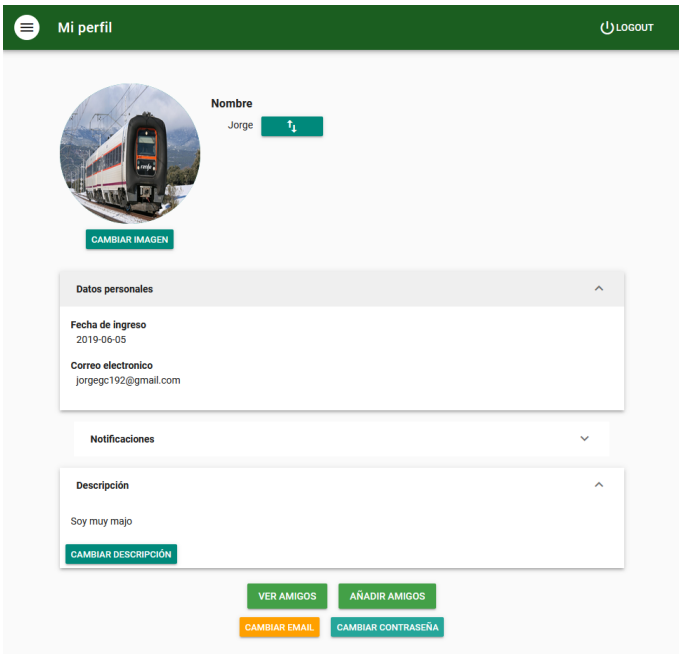


Figura 4.1: Lista de juegos en la versión móvil

más adelante se reutilizarán en las siguientes iteraciones.

Por otra parte de las clases que acogen las funcionalidades básicas empiezan a desarrollarse aquí. Las clases principales del proyecto son Usuarios, Juegos y Reuniones. Se implementaran los métodos mas básicos para poder interactuar mínimamente con la combinación de la interfaz.

Respecto a la parte servidor, correspondiente al modelo, se implementarán los métodos que establecen los datos dentro del base de datos.

El tiempo aproximado de esta iteración seria 6 semanas. Cabe añadir que hay requisitos no previstos añadidos en esta iteración, que son la visualización de los perfiles de juegos y reuniones, que se obviaron durante la fase de análisis.

4.2.2. Iteración 2: Ampliación de funcionalidades de usuarios

Los requisitos funcionales afrontados en durante esta iteración:

1.1.7	1.1.19	1.2.2	1.4.2	1.4.5
1.1.12	1.1.21	1.3.1	1.4.3	1.4.6
1.1.13	1.2.1	1.4.1	1.4.4	1.4.7

Durante esta iteración se focalizará el esfuerzo en construir las funcionalidades para el usuario, centradas en interactuar con otros objetos del proyecto. Se volverán a reutilizara las piezas visuales de filtro para buscar usuarios. Se desarrollará el componente visual para la elección y presentación de la ubicación de una reunión.

Dentro de la vista-modelo, simplemente se extenderán las clases para dará cabida a los nuevos requerimientos de esta iteración. Las clases que sufrirán cambios son Usuario y Reuniones.

Respecto al modelo, se comenzaran a utilizar las funciones de integración propias de Firebase Database en la vista-modelo. Esta implantación conllevará tiempo, ya que sera necesario implementarlas para cada

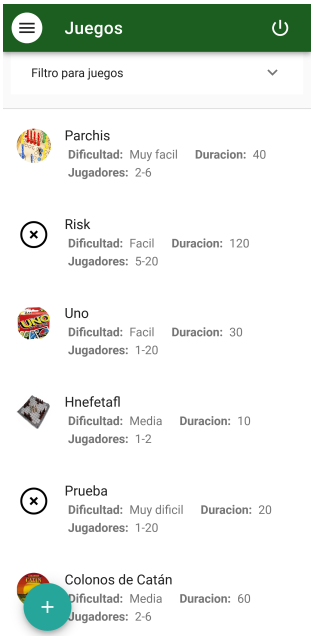


Figura 4.2: Lista de juegos en la versión móvil

objeto creado y que ha de guardarse en la base de datos.

El tiempo aproximado de implementación de esta iteración serán 4 semanas.

4.2.3. Iteración 3: Ampliación de funcionalidades de reunión y evaluación. Aplicación en Android

Los requisitos funcionales afrontados en durante esta iteración:

1.1.14	1.3.2	1.3.5	1.3.8	1.7.2
1.1.17	1.3.3	1.3.6	1.3.9	1.7.3
1.3.1	1.3.4	1.3.7	1.7.1	

En esta parte se afronta la implementación el formulario de evaluación y la visualización de la evaluación de una reunión. La interfaz debe mostrar la posibilidad de evaluar, o no, y mostrar el contenido de la evaluación en función del estado de la reunión. También se debe implementar el uso de la ubicación de una reunión que se establece cuando es creada.

Es momento de añadir la clase evaluación en la vista-controlador. Por otra parte, se han implementado los cambios a tiempo real para mantener la interfaz siempre actualizada. A parte de los requisitos funcionales, también se ha implementado el acceso correcto a la aplicación, no permitiendo que un usuario no conectado pueda acceder a partes de la aplicación que solo pueden acceder usuarios registrados.

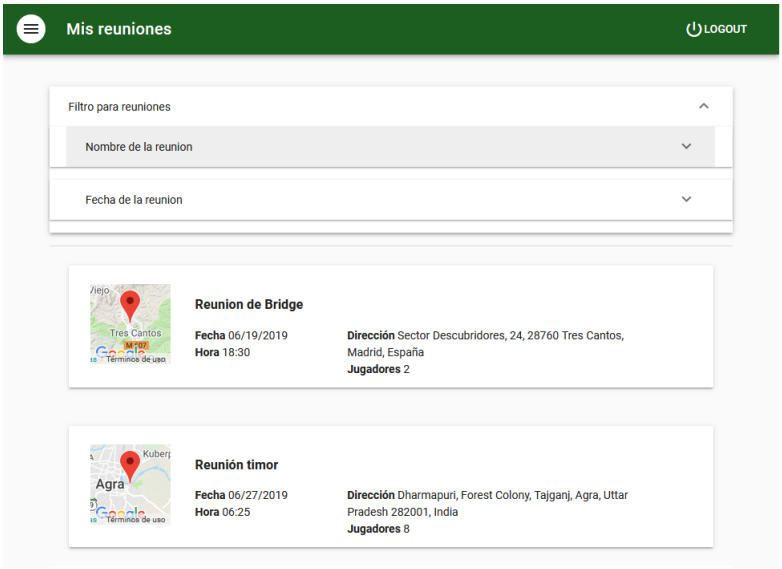


Figura 4.3: Filtro y lista de reuniones en la versión web

Otro paso importante y crucial del desarrollo es la construcción de la aplicación para Android con Cordova. Esto incrementará el tiempo de iteración. El tiempo previsto para esta parte es de 6 semanas.

4.2.4. Iteración 4: Notificaciones

Los requisitos funcionales afrontados en esta iteración:

2.2.4	2.2.6	2.2.8
2.2.5	2.2.7	

Los requisitos no funcionales afrontados en esta iteración: Mostrar notificaciones cuando se crea un juego, se ha evaluado una reunión, se el usuario sea añadido como amigo, se el usuario sea añadido a una reunión y cuando quede un día para la reunión.

A diferencia de las iteraciones anteriores, se requiere un mayor trabajo en los servicios del proveedor Firebase, implementando de esta forma las funciones encargadas de generar las notificaciones. Se añadirá durante esta iteración la clase Notificación, por lo tanto, es necesario crear la parte visual en la vista y su clase en la vista-modelo, y a su vez, con la integración con la base de datos que mantiene el modelo. Por otro lado, se debe implementar en Android las notificaciones en segundo plano.

También se debe implementar las funciones que se encarguen de controlar los cambios dentro del sistema para mantener la interfaz actualizada en la versión Android.

4.2.5. Iteración 5: Añadir clase grupos

Los requisitos funcionales afrontados en esta iteración:

Se debe implementar una nueva clase, con sus métodos para la creación y modificación. Como es obvio, también hay que trabajar en la parte visual para estos nuevos requisitos. Lo mas importante para esta clase es su interacción con las demás, es decir, la creación de una reunión con los participantes del grupo y la gestión de acceso a los grupos.

4.2.6. Iteración 6: Añadir clase espacios

Los requisitos funcionales afrontados en esta iteración:

De nuevo, se debe implementar una nueva clase que solicita la creación de los espacios, el perfil de espacio y las modificaciones de sus atributos, implicando en ello todas sus componentes visuales. No se espera que ningún cambio necesario dentro del modelo, mas que añadir el nuevo objeto implementado.

Se espera un tiempo aproximado de 2 semanas para implementar esta nueva clase completamente.

5

Integración, pruebas y resultados

5.1. Integración de la aplicación en distintas plataformas

Dado que el proyecto se ha planteado de manera incremental, no se ha desarrollado hasta la realizar todas las iteraciones descritas en el capitulo anterior. El proyecto se ha desarrollado hasta la cuarta iteración, abarcando las integraciones y pruebas llegadas hasta este punto.

La integración no ha supuesto muchas dificultades para el proyecto en ambas plataformas, en muchas partes de la aplicación el resultado ha sido el esperado. Pero más allá del funcionamiento en general de la aplicación, hay que comprobar el correcto funcionamiento de las características implementadas, más concretamente de los requisitos no funcionales.

Ya que el objetivo de unas pruebas de software es demostrar que se cumplen los requisitos definidos al cliente y encontrar situaciones donde el software se comporte de una forma no esperada.

Aplicación Web

La integración y las pruebas han sido mucho más sencillas en el desarrollo de la versión web. Esto se debe a que Node.js, permite compilar el código rápidamente y sin detener el servidor que provee la página web en local, dentro del ordenador del desarrollador. Ha medida que se introducen los cambios en el código, se pueden visualizar rápidamente en el navegador. Dado que los navegadores permiten la imitación de una pantalla de un móvil, se ha podido comprobar a la vez la compatibilidad de la interfaz entre móvil y web.

Aplicación móvil

La integración en dispositivos Android es mas complicada. En primer lugar, no permite ver los cambios ”en caliente” como sucedía con Nuxt, por lo tanto cada cambio realizado es necesario construir la aplicación con Nuxt y posteriormente ser traducida por Cordova.

En la sección de resultados se describen los errores encontrados durante esta sección.

5.2. Pruebas realizadas

El objetivo de las pruebas es intentar demostrar que la aplicación desarrollada durante el proyecto hace lo que se intenta que haga. Consiste en buscar errores, anomalías o información de atributos no funcionales del programa. Básicamente, los principales objetivos de las pruebas son dos:

- Demostrar al cliente, y en su efecto también al desarrollador, que se han cumplido los requisitos descritos durante el diseño. Es decir, que la aplicación es útil en la realidad.
- Encontrar situaciones donde el software se comporte de forma anómala.

De todas formas, las pruebas no garantizan que la aplicación no tenga defectos, siempre hay que esperar más posibles errores y tan solo se pueden llevar medidas para resolver, arriesgando también en parte, la aparición de nuevos problemas.

El proceso de pruebas se llevado en dos tipos distintos de categorías. Una de ellas son las pruebas de desarrollo realizadas para comprobar las implementaciones de los requisitos a medida que creaba el proyecto. La segunda son las pruebas de usuario, realizadas con personas ajenas al desarrollo, que comprobarán la experiencia de uso de la aplicación.

Dentro del tipo de pruebas, faltarían por nombrar las pruebas de unidad. Estas no se han realizado en este proyecto dado que consumirían demasiado tiempo crear un test automático.

5.2.1. Pruebas de desarrollo

Las actividades llevadas en esta sección han sido orientadas al comportamiento de los componentes creados que dan lugar en conjunto al sistema, y posteriormente la comprobación del sistema en conjunto. A medida que se implementaban los requisitos, eran comprobados después de su implementación. Esto quiere decir que esta parte de las pruebas se centrará en comprobar el funcionamiento de los requisitos.

Estas pruebas se puede dividir en dos tipos:

Pruebas de componente

El fin este tipo de pruebas es comprobar el funcionamiento correcto de los componentes que encapsulan uno, o varios, procedimientos y métodos de clases, en conjunto, ya que si se tratara de realizar de forma individual, no existirá el tiempo suficiente. Este tipo de pruebas se han dado sobre todo en la primea iteración que ha supuesto la mayor carga de trabajo en este aspecto.

El objetivo era comprobar la correcta comunicación entre la vista, la vista-modelo y el modelo, es decir entre los componentes Vue, Vuex y la base de datos de Firebase. En la primera iteración

solo se han comprobado las dos primeras, pues aún no se han implementado la correspondencia con Firebase a excepción de la autenticación. Posteriormente, en el resto de iteraciones si se ha realizado algún cambio en estos componentes de la interfaz, o se ha creado uno nuevo, se han realizado las mismas pruebas.

Para explicar, desde un punto de vista más cercano al usuario, sobre lo que significa cada componente, se puede decir que son aquellos elementos visuales que permiten interacción con el sistema ,como un filtro, realizar una modificación de un elemento o los campos que constituyen un formulario o que visualizan datos.

Pruebas de sistema

Durante el desarrollo al final de cada iteración, se ha realizado una prueba de sistema para comprobar funcionamiento correcto de todo el conjunto del sistema software. No solo al final de cada iteración, también se han tenido que hacer cambios importantes no previstos, como la migración a otro tipo de componente externo o algún cambio en el diseño.

Iteración 1

Se comprobaron el correcto funcionamiento entre la vista y la vista modelo, que significa que las interfaces gráficas se corresponden con los datos de Vuex. Dado que se han implementado las primeras clases del proyecto, pero la interacción con ellas es solo de muestra de información, las pruebas de sistema se han basado que la interfaz se comporte correctamente entre la versión móvil y la versión web.

Respecto a la integración con otros sistemas, se comprobó que la creación y el acceso de usuarios es correcta. Los datos de usuario se guardan en el modelo, Firebase Database, y los datos de autenticación se han creado en Firebase Authentication.

Iteración 2

Se verificó la integración del modelo en Vuex, es decir, la integración de Firebase en el sistema. Además de los requisitos añadidos, la prueba más importante es que los cambios que el usuario realiza en la interfaz se guardan finalmente en la base de datos de la forma esperada. Por lo tanto por cada prueba de componente existente hasta ahora en el sistema debía incluir también la comprobación del modelo.

Iteración 3

Como en iteraciones anteriores, se comprobó las nuevas funcionalidades implementadas, en este caso todo aquello relacionado con las evaluaciones. Se llevó a cabo las pruebas de componentes

necesarias para garantizar el correcto funcionamiento y visualización de las evaluaciones de reuniones. Otra característica importante del sistema que se ha implementado en esta iteración ha sido la recarga de los datos a tiempo real. Por lo tanto, se ha comprobado en todo el sistema si la actualización de los componentes de la interfaz es correcta cuando se modifica un juego o reunión. Otra integración importante ha sido la del plugin de Vue, vue2-google-maps, para la visualización y edición de ubicación de mapas de Google Maps. Era necesario asegurar que los datos proporcionados por el plugin eran formateados correctamente y que posteriormente sirvieran para la muestra de la localización de las reuniones. Estos datos son calle, ciudad o municipio, país y las coordenadas geográficas.

Figura 5.1: Añadir jugador en una reunión dentro de la web

Aparte de las nuevas funcionalidades, han sido necesarios cambios en la vista-controlador y en el modelo. En el primero, la estructura de los ficheros de Vuex fue modificada, por recomendación del framework base Nuxt en su nueva versión, no supuso dificultades. No sucedió lo mismo para el modelo, ya que se ha migrado de Firebase Database a Firebase Firestore por facilitar la implementación de los cambios a tiempo real. Por lo tanto, se ha realizado una única para la comprobación de cambios a tiempo real y la integración de la nueva base de datos.

Una de las partes de pruebas de integración del sistema más importante ha sido la compilación de la aplicación a Android con Apache Cordova. De nuevo fue necesario repasar todas las partes de la aplicación desde un móvil actual real. Todas las pruebas se han realizado desde un Xiaomi 4X con una versión de Android 7.1.2 en conjunto con las herramientas de desarrollo de Google Chrome.

Iteración 4

En esta iteración se desarrolló la clase notificación, con una implementación gráfica de sencilla de Snackbars para ambas plataformas y las notificaciones en segundo plano móviles para Android. Por lo tanto, en primer lugar se empezó con la prueba de integración de las notificaciones y la correcta visualización de las Snackbars. Ha sido importante asegurar que las notificaciones se generaban correctamente y que el enlace entre vista y vista-modelo funcionara sin un retraso considerable (no mas de 10-20 segundos) desde que se realiza el cambio hasta que se notifica, estando en primer plano la aplicación Web y Android, y solamente en segundo plano en Android.

Un problema relevante de esta iteración ha sido la dificultad de añadida de un plugin desactualiza-

do de Apache Cordova que mantenía el enlace con el proveedor de servicios de Firebase. El plugin cordova-plugin-firebase dejo de mantenerse en su repositorio habitual dos semanas antes de comenzar la aplicación de este servicio. Hasta que se migro a su repositorio actualizado de cordova-plugin-firebase-lib.

5.2.2. Pruebas de usuario

El objetivo de estas pruebas es observar como se comportaría la aplicación en un entorno natural, con usuarios creando contenidos e interactuando con ellos dentro de la aplicación y como interactúan entre ellos. Con el fin de realizarlas, se ha pedido a un grupo de personas que prueben la aplicación, tanto en la versión móvil como en la versión web, durante 2 semanas y traten de simular el comportamiento de usuarios reales de la aplicación proporcionando una versión beta de la aplicación.

Esto supone los probadores comprueben los requisitos funcionales, para comprobar si la aplicación cumple los objetivos mínimos propuestos en la seccion de diseño. También se tendrán en cuenta los requisitos no funcionales como las notificaciones, tiempos de carga o intuitividade y facilidad de uso de la aplicación.

Para recoger estos datos se ha realizado una encuesta en la que se puntúa un unas características concretas que abarcan los objetivos mínimos y los casos de uso definido. El resultado y una descripción mas profunda de las pruebas de usuario se puede ver en el anexo.

6

Conclusiones y trabajo futuro

6.1. Conclusión

El objetivo del proyecto era crear una red social que ponga en contacto personas que disfruten de los juegos de mesa. Con el fin de alcanzar este objetivo se han pasado por distintas fases, tanto de creación como de auto aprendizaje.

La elaboración del análisis y diseño del proyecto ha permitido tomar importancia de la relevancia de estas fases. Si se mantiene este criterio, se puede apreciar que el análisis ayuda a conocer que va ser necesario, como que herramientas software utilizar, y el diseño a como planear como afrontar los requerimientos que sean necesarios, mientras que a su vez, se prueba e integra la aplicación en construcción.

Respecto al los modelos de desarrollo software utilizados, se pueden observar como han evolucionado para proporcionar orientación a herramientas mas versátiles, que tratan de llevar la lógica fuera del modelo para facilitar una implementación mas sencilla de una parte servidor y llevando mayor carga de desarrollo al backend e interfaz de una aplicación.

A su vez las tecnologías de creación de aplicaciones, sobre todo aquellas que son híbridas, ha revolucionado las capacidades de los desarrolladores. Pero que también utilizar tecnologías demasiado modernas pueden que hagan que los errores no sean tan fáciles de resolver o que no haya tanto apoyo de la comunidad.

Aun así, todo esto a servido para experimentar con tecnologías y modelos que no se han visto durante el grado, y como se afrontaría un proyecto desde cero.

6.2. Trabajo futuro

En primer lugar, el proyecto esta planteado de forma iterativa y se alcanzado la iteración 4, faltando por desarrollar las iteraciones 5 y 6. Por otra parte, se deben corregir los errores encontrados durante las pruebas de usuario, como

Fuera de las características fuera del proyecto recogido, seria algunas muy útiles a a añadir que mejorarían la experiencia.

- Añadir un sistema de recomendación que permita al sistema interactuar con el usuario, ofréndenles juegos, reuniones o usuarios que pudieran interesar o sentir afinidad.
- Aunque se decidió no implementarlo durante el análisis, un sistema de mensajería, o chat, de la red social resultaría útil para los usuarios.
- La integración de usuarios que no fueran personas sino entidades. Es decir, una tienda que permita publicitarse o ceder su espacio al resto de usuarios.
- Permitir la subida de ficheros de un dispositivo. Resultaría muy útil sobre todo para los usuarios de aplicaciones móviles, ya que es común entre las aplicaciones móviles este tipo de característica.

Referencias

1. Manuel Gutiérrez, *"Los adictos al móvil no paran de crecer pese al aumento en el precio de las tarifas"*, 14 de julio, El Mundo, 2018, Madrid

2. Antonio Terán Prieto, *"Ciberadicciones. Adicción a las nuevas tecnologías (NTIC)"*, 16º Congreso de Actualización de Pediatría, Febrero, 2019, Palencia

3. Irene Seguranyes *"Beneficios de los juegos de mesa para la familia", Edukame.com, Visto en junio 2019*

4. Timpik, aplicacion red social y deportiva: http://www.timpik.com/

5. Iván Ramírez, "Historia y evolución de Android: cómo un sistema operativo para cámaras digitales acabó conquistando los móviles", Xataka.com, Marzo, 2019.

6. *"Android"*, 12 de diciembre, Blog Museo de la Informática, Universidad Politécnica de Valencia, 2014 Valencia

7. Borja Simancas Delgado, *"10 años de Android: así ha evolucionado el mejor sistema móvil"*, 23 de septiembre, Diario El Español Android Libre, 2018

8. Webview para Android:
   https://developer.chrome.com/multidevice/webview/overview

9. Andre Charland, Brian Leroux, *"Mobile Application Development: Web vs. Native"*, Communications of the ACM, Mayo, 2011

10. Anthony I. Wasserman, *"Software Engineering Issues forMobile Application Development"*, Carnegie Mellon Silicon Valley, visto en junio 2019

11. Alex Holderness, *"Is the Angular Decline a Myth?"*, 14 de febrero, Hackernoon.com, 2018

12. Max Lynch, *"Introducing Ionic 4: Ionic for Everyone"*, 23 de enero, Blog Ionic Framework, 2019

13. Tom Occhino, *"React Native: Bringing modern web techniques to mobile"*, 26 de marzo, Code Fb, 2015

14. Visual Studio Tools para Xamarin:
    https://visualstudio.microsoft.com/es/xamarin/"

15. 1 Node.js:
    https://nodejs.org/es/

16. Vue.js :
    https://vuejs.org/

17. Nuxt.js:
    https://nuxtjs.org/

18. Vuetify.js :
https://vuetifyjs.com/es-MX/

19. Vuex:
https://vuex.vuejs.org/

20. Google Firebase:
https://firebase.google.com/

21. Apache Cordova:
https://cordova.apache.org/

Anexos

6.3. Escenarios de casos de uso

6.3.1. Registro de usuario

Identificador: 000-000	**Nombre:** Registro de usuario
Autor: Jorge Gómez	**Elementos involucrados:** Usuario
Resumen: Un nuevo usuario navega hasta la pagina web o la aplicación. Uno de los apartados principales o en el apartado de Login mostrara la opción para crear un nuevo usuario. Se presentara un formulario que solicitara la información requerida para el registro **Pre-condiciones:** El usuario debe tener un correo electrónico **Post-condiciones:** Los datos proporcionados deben guardarse en la base de datos	

Sucesión de eventos	
Usuarios	Sistema
1. Acceder al formulario de registro 2. Introducir datos solicitados 3. Finalizar formulario	1. Comprobar datos de entrada 2. Registrar usuario en la base de datos

Problemas y/o alternativas
Elementos involucrados: Ninguno adicional
Dar posibilidad de registro con la cuenta de Google.

000-

6.3.2. Login de usuario

Identificador: 000-001	**Nombre:** Login de usuario
Autor: Jorge Gómez	**Elementos involucrados:** Usuario
Resumen: El usuario se dispone a acceder al sistema (desde la versión web y versión móvil). Se requiere para el acceso usuario y contraseña. **Pre-condiciones:** El usuario se ha registrado correctamente. **Post-condiciones:** Debe dar acceso a todas las funcionalidades de usuario conectado.	

Sucesión de eventos	
Usuarios	Sistema
1. Acceder al formulario de login. 2. Introducir e-mail y contraseña.	1. Comprobar e-mail y contraseña correctos.

Problemas y/o alternativas
Elementos involucrados: Usuario
Como en el registro, permitir el acceso con cuenta de Google.

6.3.3. Creación de juego

Identificador: 000-002	**Nombre:** Crear juego
Autor: Jorge Gómez	**Elementos involucrados:** Usuario, juego
Resumen: Un usuario rellenara un formulario con los datos del juego. **Pre-condiciones:** Acceso a la aplicación. **Post-condiciones:** El juego debe guardarse en la base de datos del sistema.	
Sucesión de eventos	
Usuarios	Sistema
1. Acceder al formulario de creación de juego 2. Completar formulario	1. Comprobar datos de entrada 2. Registrar juego en la base de datos
Problemas y/o alternativas	
Elementos involucrados: Ninguno/a	
Ninguno/a	

6.3.4. Creación de reunión

Identificador: 000-003	**Nombre:** Crear reunión
Autor: Jorge Gómez	**Elementos involucrados:** Usuario, juego, reunión
Resumen: Un usuario rellenara un formulario con los datos de la reunión. Los datos principales son nombre, fecha, jugadores máximos y mínimos y juegos para jugar. **Pre-condiciones:** Acceso a la aplicación. Al menos debe de existir un juego en la base de datos para añadir a la reunión. Si no hay juegos o no existe el juego que busca el usuario en la base de datos, habría que añadir alguno. **Post-condiciones:** La reunión debe guardarse en la base de datos del sistema.	

Sucesión de eventos	
Usuarios	Sistema
1. Acceder al formulario de creación de reunión 2. Completar formulario	1. Comprobar datos de entrada 2. Registrar reunión en la base de datos

Problemas y/o alternativas
Elementos involucrados: Ninguno/a
Ninguno/a

6.3.5. Búsqueda de juegos

Identificador: 000-004	**Nombre:** Buscar juego
Autor: Ejemplo	**Elementos involucrados:** Usuario, juego
Resumen: El usario se dispone a buscar juegos. Se presentaran en lista y podrán ser filtrados por nombre. **Pre-condiciones:** Acceso a la aplicación. Que haya juegos en la base de datos **Post-condiciones:** Debe mostrarse los juegos correctamente en función del filtro	

Sucesión de eventos	
Usuarios	Sistema
1. Acceder a la lista de juegos. 2. Completar formulario.	1. Mostrar juegos de la base de datos. 2. Mostrar juegos filtrados de la base de datos.

Problemas y/o alternativas
Elementos involucrados: Ninguno/a
Ninguno/a

6.3.6. Búsqueda de reunión

Identificador: 000-005	**Nombre:** Buscar reunión
Autor: Jorge Gómez	**Elementos involucrados:** Usuario, reunión
Resumen: El usuario no conoce a nadie de su entorno. Por lo tanto se dispone a buscar reuniones dentro de la aplicación. Se presentara una lista ordenada por la fecha mas cercana a la actual. Posteriormente la lista podrá ser filtrada por nombre, distancia y fecha. **Pre-condiciones:** Acceso a la aplicación. Que existan reuniones en la base de datos. **Post-condiciones:** Debe mostrarse las reuniones ordenadas correctamente en función del filtro.	

Sucesión de eventos	
Usuarios	Sistema
1. Acceder a la lista de reuniones 2. (Opt)Establecer valores del filtro	1. Obtener lista de reuniones 2. (Opt)Filtrar y ordenar dados los datos por el usuario.

Problemas y/o alternativas
Elementos involucrados: Ninguno/a
Ninguno/a

6.3.7. Unirse a reunión

Identificador: 000-000	**Nombre:** Unirse a reunión
Autor: Jorge Gómez	**Fecha:** DD-MM-AA
Elementos involucrados: Usuario, reunión	
Resumen: El usuario ha encontrado una reunión y decide introducirse en el perfil de la misma. Desde aquí podrá ver todas las características de la reunión, incluyendo juegos y jugadores que participaran. Dado que le gusta la reunión, decide unirse a ella. **Pre-condiciones:** Debe existir una reunión y que no sea privada y este abierta. **Post-condiciones:** Unirse a la reunión debe reflejarse en el registro de usuario y reunión de la base de datos del sistema.	

Sucesión de eventos	
Usuarios	Sistema
1. Acceder al sistema. 2. Acceder a la lista de reuniones y seleccionar reunión. 3. Pulsar unirse a reunión.	1. Registrar cambios en la base de datos.

Problemas y/o alternativas
Elementos involucrados: Ninguno/a
En lugar de que el usuario vaya hacia la reunión, es decir buscándola, podría existir la alternativa inversa. El sistema recomienda la reunión al usuario, por distintos factores, dándole la opción de unirse a la reunión.

6.3.8. Modificar perfil de usuario

Identificador: 000-000	**Nombre:** Ejemplo
Autor: Ejemplo	**Fecha:** DD-MM-AA
Elementos involucrados: Usuario	
Resumen: Un usuario quiere modificar cualquiera de los datos que le define y le muestra a los demás. Dentro de su propio perfil debe poder encontrar las opciones que le permitirán cambiar su perfil a su gusto. **Pre-condiciones:** Registro y acceso a la aplicación. **Post-condiciones:** Los cambios deben guardarse y permanecer en el sistema.	

Sucesión de eventos	
Usuarios	Sistema
1. Acceder al perfil propio 2. Seleccionar el dato que quiere cambiar 3. Introducir el nuevo valor y guardar	1. Registrar cambios en la base de datos

Problemas y/o alternativas
Elementos involucrados: Ninguno/a
Ninguno/a

6.3.9. Añadir jugador

Identificador: 000-000	**Nombre:** Ejemplo
Autor: Jorge Gómez	**Fecha:** DD-MM-AA
Elementos involucrados: Usuario, Reunión.	
Resumen: Un usuario a creado una reunión, pero le gustaría que en esta reunión solo participen aquellas personas que el invite. Desde el mismo perfil de la reunión, se mostrara un opción que dispondrá de una lista para seleccionar aquellos jugadores que quiera invitar. **Pre-condiciones:** Acceso a la aplicación. Reunión creada por el mismo usuario y que sea privada. **Post-condiciones:** Los cambios deben guardarse correctamente y mostrarse al usuario.	

Sucesión de eventos	
Usuarios	Sistema
1. Acceder al perfil de reunión. 2. Acceder a añadir usuarios 3. Seleccionar los jugadores a añadir	1. Comprobar datos de entrada 2. Registrar juego en la base de datos

Problemas y/o alternativas
Elementos involucrados: Ninguno/a
Mostrar aparte una lista de amigos del usuario. Por otra parte, se podría contar también con invitaciones, que esperan confirmación del usuario invitado para añadirlo a la reunión.

6.3.10. Añadir descripción

Identificador: 000-009	**Nombre:** Ejemplo
Autor: Ejemplo	**Elementos involucrados:** Usuario, juego
Resumen: El usuario ha decidido notificar un cambio en la reunión y quiere mostrarlo en la descripción de la reunión. Debe ir hasta el perfil de la reunión y desde aquí cambiarla. **Pre-condiciones:** Acceso a la aplicación. Reunion creada y ser el creador. **Post-condiciones:** Los cambios deben guardarse correctamente en el sistema.	

Sucesión de eventos	
Usuarios	Sistema
1. Acceder al perfil de reunión. 2. Acceder a cambiar descripción y introducir el nuevo valor.	1. Los cambios deben guardarse en el sistema.

Problemas y/o alternativas
Elementos involucrados: Ninguno/a
Ninguno/a

6.3.11. Evaluar reunión

Identificador: 000-000	**Nombre:** Ejemplo
Autor: Ejemplo	**Fecha:** DD-MM-AA
Elementos involucrados: Usuario, Reunión, Formulario.	
Resumen: El usuario creador se dispone a evaluar la reunión. Desde el perfil de la reunión se mostrará un formulario para evaluar la reunión, donde los usuarios acordaran quienes han sido los ganadores de cada juego al que se ha jugado en la reunión. También pueden anotar alguna incidencia relacionada con alguno de los jugadores durante la reunión. **Pre-condiciones:** Acceso a la aplicación. **Post-condiciones:** El juego debe guardarse en la base de datos del sistema.	

Sucesión de eventos	
Usuarios	Sistema
1. Acceder al formulario de creación de juego 2. Completar formulario	1. Comprobar datos de entrada 2. Registrar juego en la base de datos

Problemas y/o alternativas
Elementos involucrados: Ninguno/a
Ninguno/a

6.3.12. Recomendación

Identificador: 000-011	**Nombre:** Ejemplo
Autor: Jorge Gómez	**Elementos involucrados:** Usuario A, Usuario B, Reunión
Resumen: En este caso el usuario espera al aviso del sistema. Un usuario A crea una reunión con unos parámetros que resultan favorablemente interesantes para un usuario B (podría ser bien por la distancia de la reunión, los juegos a los que se van a jugar o usuarios conocidos). El sistema notificara al usuario B la nueva reunión del usuario A. **Pre-condiciones:** Al crearse la reunión por A, debe mantener unas características y condiciones que el sistema al evaluarlas, pueda recomendarlas al usuario B. **Post-condiciones:** Debe mostrarse al usuario B y que de forma directa navegue hasta el perfil de la reunión.	

Sucesión de eventos	
Usuarios	Sistema
	1. Evaluar reunión recién creada. 2. Si el resultado es positivo, mostar mensaje a usuario B.

Problemas y/o alternativas
Elementos involucrados: Ninguno/a
En lugar de evaluar las reunión recién creada, podrían mostrase las recomendaciones cuando el usuario acaba de conectarse y pueda recibir todas los notificaciones de una sola vez.

6.4. Maquetas de interfaz de usuario

6.4.1. Marco principal de ventana

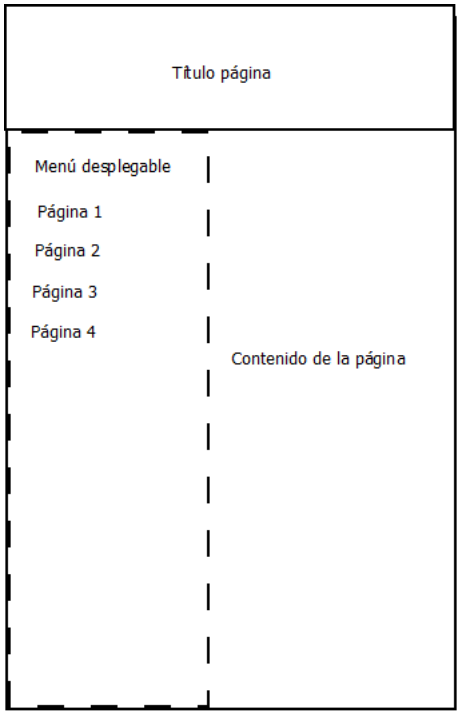


Figura 6.1: Maqueta de la disposición del marco en dispositivos móviles.

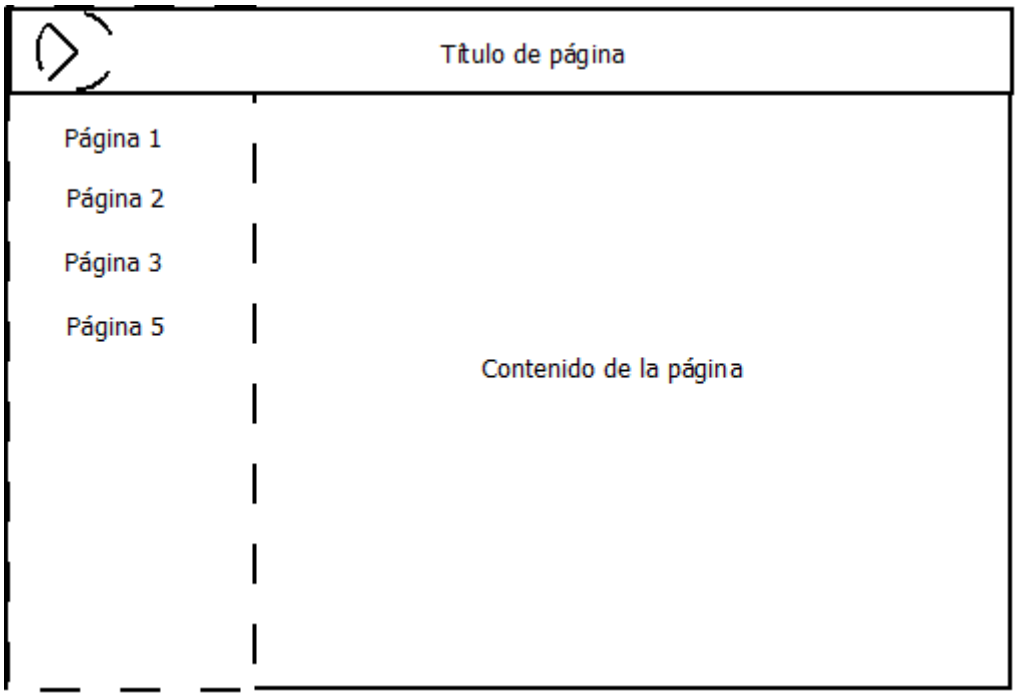


Figura 6.2: Maqueta de la disposición del marco en aplicación web.

6.4.2. Perfil de usuario

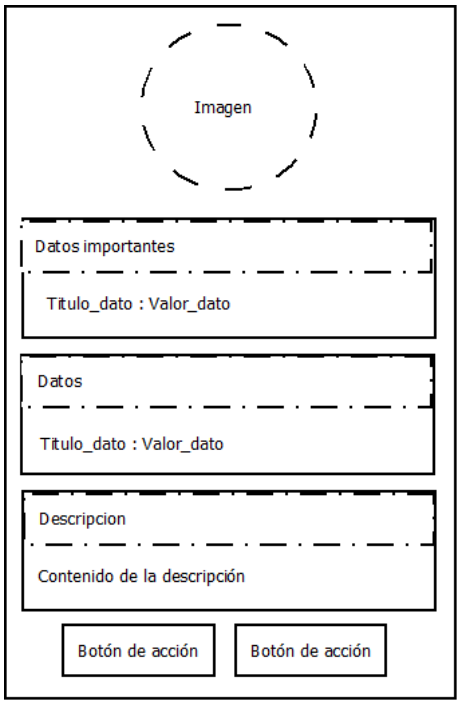


Figura 6.3: Maqueta del perfil de usuario en dispositivos móviles.

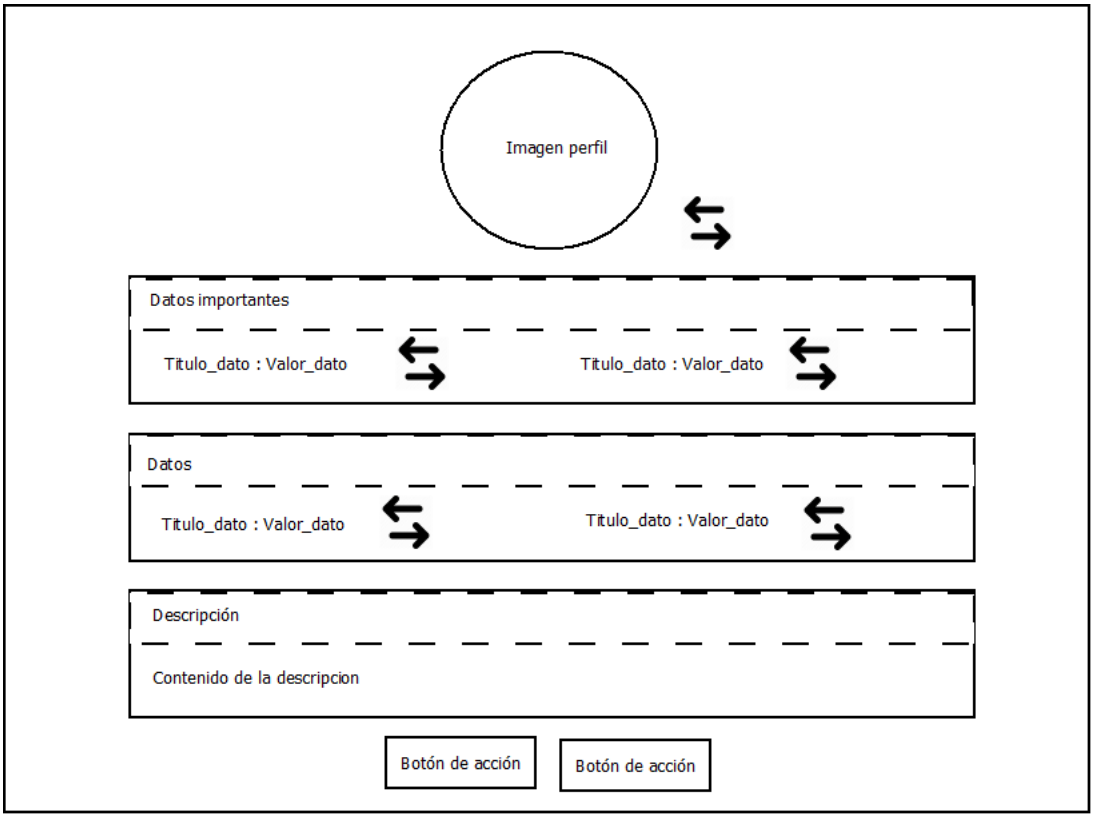


Figura 6.4: Maqueta del perfil de usuario en aplicación web.

6.4.3. Lista de elementos

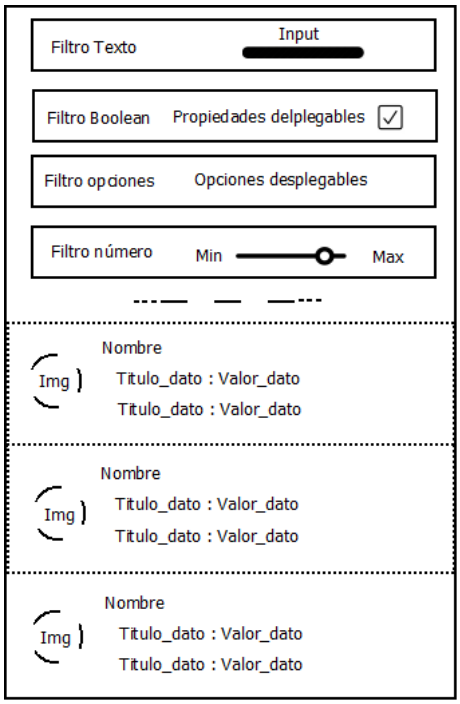


Figura 6.5: Maqueta de la disposición elementos en lista con filtro en dispositivos móviles.

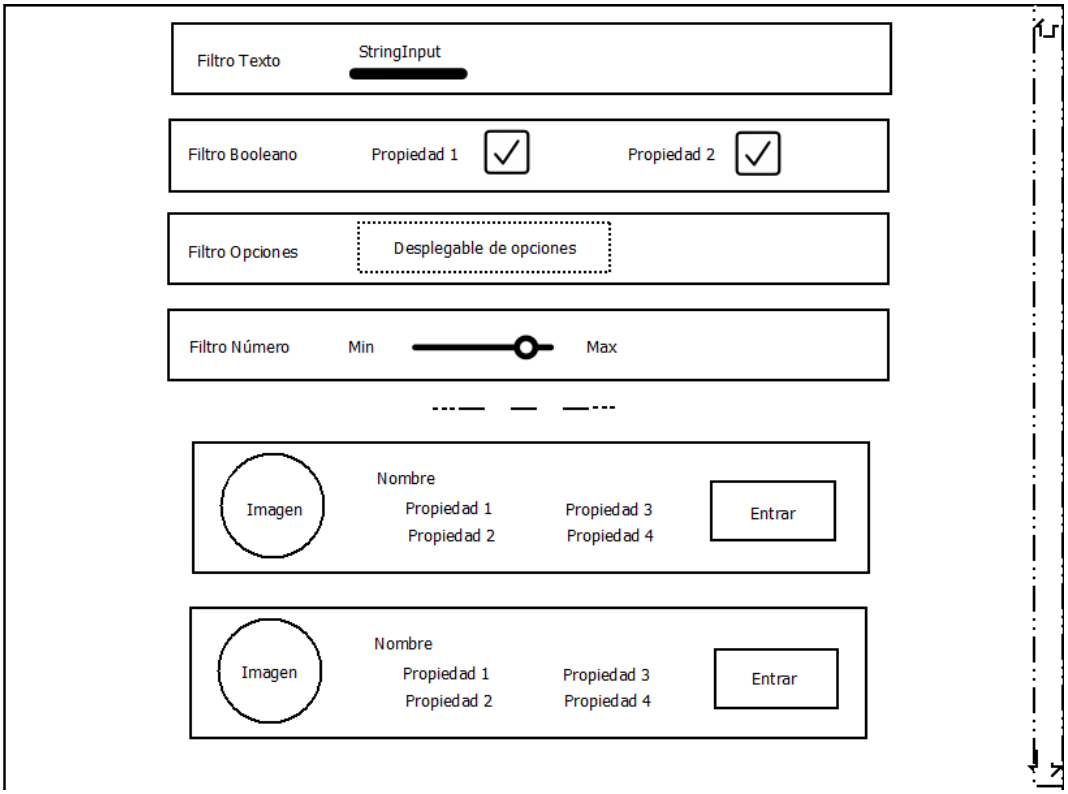


Figura 6.6: Maqueta de la disposición elementos en lista con filtro en aplicación web.

6.4.4. Login

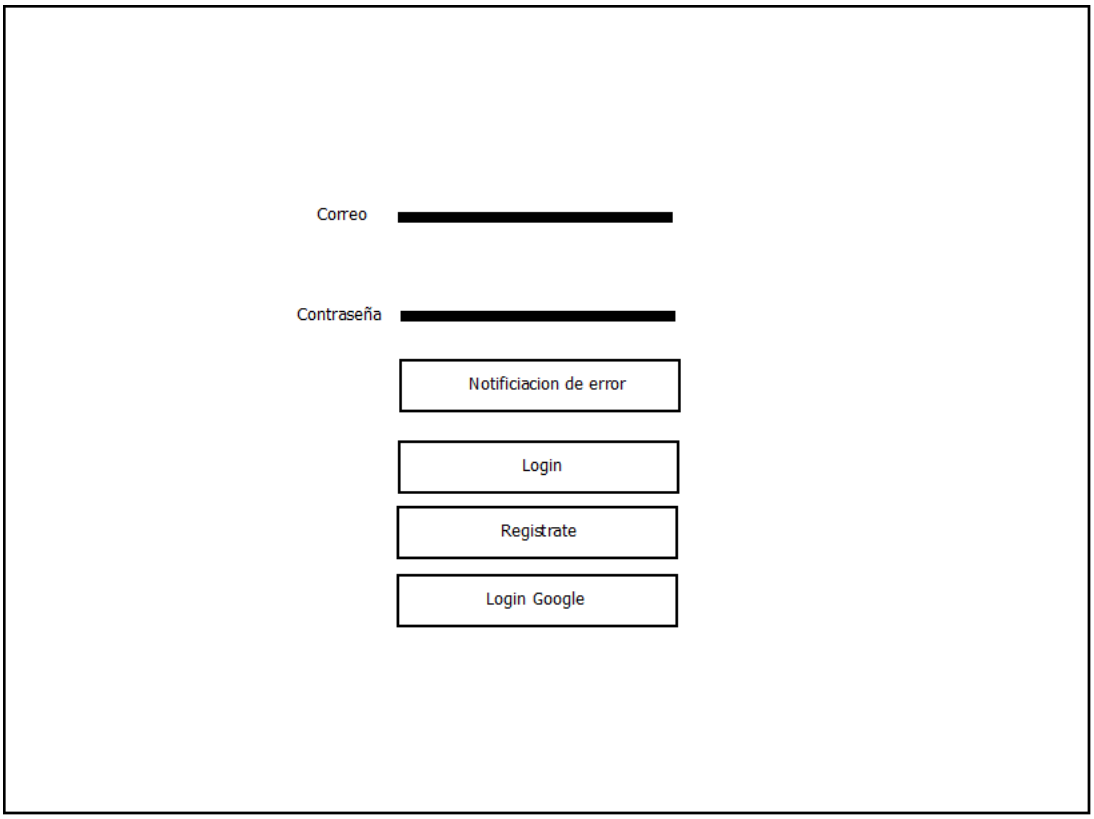


Figura 6.7: Maqueta de loggin en aplicación web.

6.4.5. Menu principal

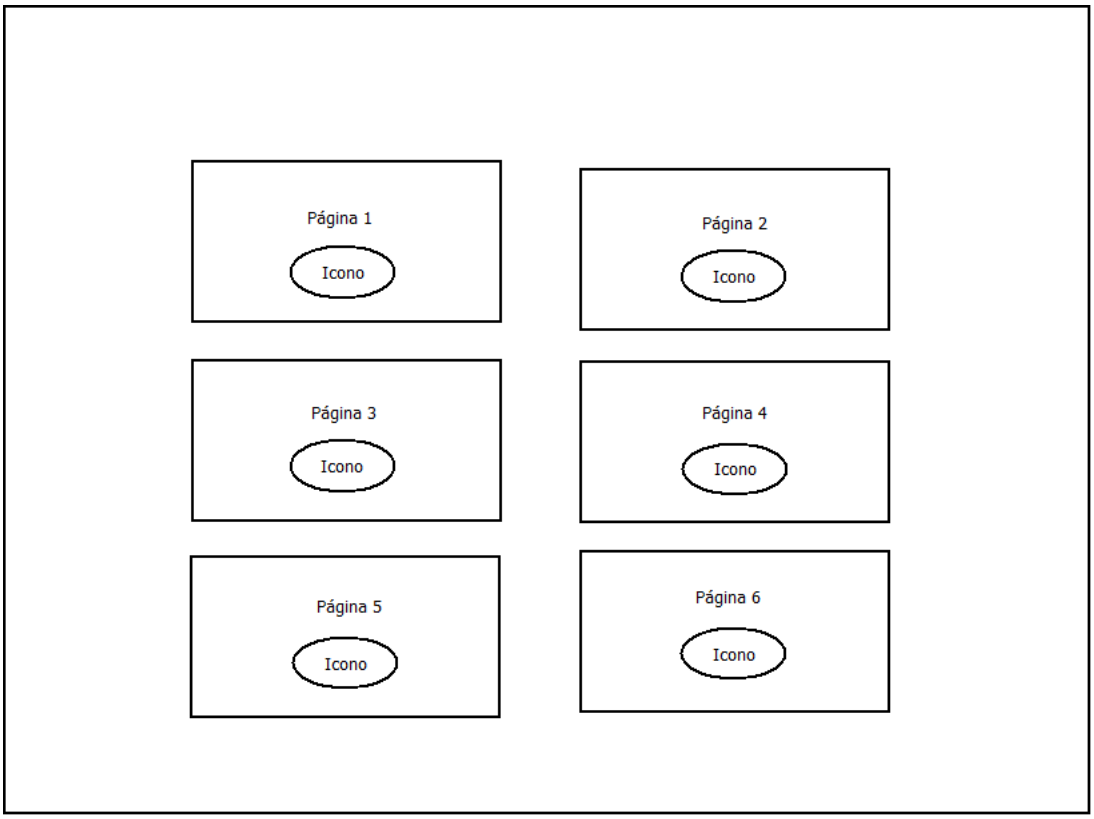


Figura 6.8: Maqueta de la disposición del menú principal en aplicación web.

6.5. Requisitos del proyecto

En esta sección se enumeran todos los requisitos licitados durante la fase de análisis.

6.6. Requisitos funcionales

6.6.1. Requisitos funcionales de todos los usuarios

1.1.1 Crear cuenta de usuario.

1.1.2 Crear y borrar reunión.

1.1.3 Crear juegos.

1.1.4 Crear grupo

1.1.5 Crear ubicación

1.1.6 Identificarse para el acceso a la red social.

1.1.7 Modificar perfil de usuario.

1.1.8 Unirse a reuniones.

1.1.9 Salir de reuniones.

1.1.10 Unirse a un grupo.

1.1.11 Salir de un grupo.

1.1.12 Marcar juego como favorito

1.1.13 Desmarcar juego como favorito

1.1.14 Ver reuniones pasadas.

1.1.15 Buscar reunión por nombre.

1.1.16 Buscar reunión por fecha.

1.1.17 Buscar reunión por distancia.

1.1.18 Buscar juego por nombre.

1.1.19 Buscar juego por duración.

1.1.20 Buscar juego por dificultad.

1.1.21 Buscar juego por tipo.

1.1.22 Buscar grupo por nombre.

1.1.23 Ver tablón de notificaciones

1.1.24 Borrar notificaciones.

6.6.2. Interacción entre usuarios

1.2.1 Agregar usuario como amigo

1.2.2 Borrar usuario como amigo

6.6.3. Interacción con reuniones como administrador

1.3.1 Añadir o borrar usuario de la reunión.

1.3.2 Añadir juegos a la reunión.

1.3.3 Establecer numero máximo y mínimo de jugadores para la reunión.

1.3.4 Establecer reunión abierta o cerrada para cualquier usuario

1.3.5 Establecer reunión privada.

1.3.6 Establecer ubicación de reunión.

1.3.7 Establecer hora y fecha de la reunión.

1.3.8 Establecer descripción de la reunión.

1.3.9 Editar características de la reunión.

6.6.4. Interacción con juegos como administrador

1.4.1 Editar jugadores máximos y mínimos del juego.

1.4.2 Editar nombre del juego.

1.4.3 Editar imagen del juego.

1.4.4 Editar tipo de juego.

1.4.5 Editar tiempo de juego.

1.4.6 Editar dificultad.

1.4.7 Editar descripción de juego.

6.6.5. Interacción con grupos como administrador

1.5.1 Editar nombre de grupo.

1.5.2 Editar ubicación de grupo.

1.5.3 Admitir o denegar peticiones de acceso al grupo.

1.5.4 Borrar usuarios del grupo.

6.6.6. Interacción con espacios como administrador

1.6.1 Editar nombre del espacio.

1.6.2 Editar ubicación de espacio.

1.6.3 Editar si es público o privado.

6.6.7. Interacción con evaluaciones como administrador

1.7.1 Establecer ganadores

1.7.2 Establecer incidencias

1.7.3 Establecer descripción.

6.7. Requisitos no funcionales

6.7.1. Seguridad

2.1.1 No permitir el acceso a usuarios no registrados.

2.1.2 No permitir la modificación de elementos en el que el usuario no es el administrador.

6.7.2. Operacional

2.2.1 La aplicación debe poder utilizarse tanto en versión Android como en versión Web.

2.2.2 Los cambios que se realicen en ambas plataformas deben mostrarse en ambas plataformas.

2.2.3 Mostrar notificación cuando se crea un juego nuevo.

2.2.4 Mostrar notificación cuando se cambia una reunión perteneciente.

2.2.5 Mostrar notificación cuando se ha evaluado una reunión.

2.2.6 Mostrar notificación cuando se te añadan como amigo.

2.2.7 Mostrar notificación cuando se añada una reunión al usuario.

2.2.8 Mostrar notificación cuando quede un día para la reunión.

6.7.3. Mantenimiento

2.3.1 Modularidad en el los componentes visuales.

2.3.2 Reusabilidad en el los componentes visuales.

6.7.4. Interfaz y usabilidad

2.4.1 La interfaz debe ser sencilla a la hora de mostrar datos y priorizar aquellos que sean más importantes.

2.4.2 Mostrar gráficamente las ubicaciones de las reuniones y los espacios.

2.4.3 Toolbar superior (básica para dispositivos móviles)

2.4.4 Barra de navegación lateral (básica para dispositivos móviles)

6.8. Ejemplo de código de desarrollo

6.9. Vuetify

A continuación se hace muestra de un fragemento de codigo de Vuetify.

```
<template>
  <v-container>
    <v-card class="pa-2" @click="onClick">
      <v-layout row wrap pa-1>
        <!-- Parte izquierda para imagen y puntuacion-->
        <v-flex xs2 mb-3>
          <v-layout column wrap
            class="justify-center align-center pt-3">
            <!--Lugar para la imagen-->
            <v-avatar size="90" color="grey">
              <slot/>
            </v-avatar>
            <v-rating
              v-if="rating"
              v-model="ratingRate"
              small
              readonly/>
          </v-layout>
        </v-flex>

        <!-- Parte derecha para nombre y propiedades-->
        <v-flex xs9>
          <v-layout column>
            <!-- Nombre -->
            <v-card-title primary-title>
              <h3>{{ name }}</h3>
            </v-card-title>
            <v-layout row wrap ml-3>
              <!-- Columna izquierda -->
              <v-flex v-if="propertiesLeftRow" xs4>
                <div v-for="prop in propertiesLeftRow"
                  :key="prop.id">
                    <b>{{ prop.name }}</b>
                    {{ prop.value }}
                </div>
              </v-flex>
              <!-- Columna derecha -->
              <v-flex v-if="propertiesRightRow" xs8>
                <div
                  v-for="prop in propertiesRightRow"
                  :key="prop.id">
                    <b>{{ prop.name }}</b>
                    {{ prop.value }}
                </div>
              </v-flex>
            </v-layout>
          </v-layout>
        </v-flex>
```

```
49
50        </v-layout>
51      </v-card>
52    </v-container>
53  </template>
```

Este código corresponde al componente visual *ItemList.vue* de un elemento de una lista, ya sea juego, reunión o usuario. Para comprenderlo mejor, es necesario ver el resto del fichero, que esta compuesto por Vue.

El siguiente fragmento de código corresponde a la pagina que se encarga de contener la lista de juegos, *games/index.vue*. Se puede observar la llamada al componente anterior, mientras itera los juegos que se mostrarán por pantalla.

```
1     <!--Componente lista PC -->
2     <div class="hidden-sm-and-down">
3       <v-flex
4         v-for ="game in games"
5         :key="game.id">
6         <itemList
7           :name="game.name"
8           :properties-left-row="[
9             { name: 'Dificultad', value: game.difficulty },
10            { name: 'Duracion', value: game.timePlay }]"
11          :properties-right-row ="[
12            { name: 'Jugadores', value: game.minPlayers +'-'+ game.maxPlayers },
13            { name: 'Tipo', value: game.type }]"
14          text-button="Ir al juego"
15          :route-button="'/games/'+game.id">
16          <img v-if="game.image" :src="game.image">
17          <img v-else :src="imageNull">
18        </itemList>
19      </v-flex>
20    </div>
```

6.10. Vue

La continuación del fichero *ItemList.vue*. Al corresponder a un componente visual, no mantiene ninguna lógica respecto con el modelo. Una parte importante de los componentes Vue, es el objeto *props*, ya que permite recibir datos externos al componente visual. Todos sus valores deben ser transmitidos por un componente padre, que contiene varios componentes conformando una página.

```
<script>
  export default {
    props: {
      name: { type: String, default: "Nombre elemento", require: false },
      image:{ type: String, default: '/null_icon.png', required: false },
      rating: { type: Boolean, default: false, require: false },
      ratingRate:{ type: Number, default: 1, require: false },
      textButton: { type: String, default: 'textButton', require: false },
      routeButton: { type: String, default: '', require: true },
      propertiesLeftRow: { type: Array,
        default: () => {
          return null
        },
        required: false
      },
      propertiesRightRow: {
        type: Array,
        default: () => {
          return null//[ { name:"PropRight1", value:"Anyone"} ]
        },
        require: false
      },
    },
    methods: {
      onClick() { this.$router.push(this.routeButton) }
    }
  }
</script>
```

En cambio, la página *games/index.vue* si contiene parte de la vista-modelo y su comunicación con el modelo. Además de mostrar más partes de como esta formado un componente en Vue.

En primer lugar la definición de un *layout*, asignado a la interfaz correspondiente para un usuario registrado y una llamada a fetch, que se ocupa de cargar la lista de juegos antes de renderizar la pagina.

Pero las partes más importantes son *data*, que mantiene los datos del componente. *Computed* que se ocupa de procesar los datos que cambian, en este caso, se ocupa de cargar los juegos establecidos por el filtro. Por último, se encuentran los *methods*, y mantienen las operaciones que son llamadas por el usuario, como pulsar un botón.

```
<script>
import { mapGetters, mapActions } from 'vuex'
export default {
    layout: 'user_logged',
    async fetch ({store}) {
        await store.dispatch('games/loadGames')
    },
    data: () => {
      return{
        imageNull: require('~/static/null_icon.png'),
        nameGame: '',
        time: [0,1000],
        optionsDificulty: [
            {name: 'Muy facil', value: false},
            {name: 'Facil', value: false},
            {name: 'Media', value: false},
            {name: 'Dificil', value: false},
            {name: 'Muy dificil', value: false}
        ],
        optionsType: [
            {name: 'Cartas', value: false},
            {name: 'Tablero', value: false},
            {name: 'Wargame', value: false},
        ]
      }
    },
    computed: {
      ...mapGetters('games', ['getGamesFilter', 'getLoading']),
      games() { return this.getGamesFilter({
          name: this.nameGame,
          time: this.time,
          difficulty: this.optionsDificulty,
          type: this.optionsType
        })
      },
    },
    methods: {
        ...mapActions('games', ['fetchGameFilter']),
        updateName(name) { this.nameGame = name},
        updateTime(time) { this.time = [time[0], time[1]] },
        updateDifculty(dificulty) { this.optionsDificulty = dificulty },
        updateType(type) { this.optionsType = type }
    }
  }
</script>
```

6.11. Vuex

En esta sección se muestra el código correspondiente al modelo de Juegos. Se puede observar que se compone de las 4 partes vistas en la parte del documento 3.3. Solamente se incluyen los *getters*, *actions* y *mutations* que afectan al código anterior.

```
export const state = () => ({
 items:[],
 loading: null,
 errorGames: null,
 gameProfile: null
});

export const getters = {
 ...
 getGamesFilter: (state, commit) => filter =>{
  const name = filter.name;
  const dificulty = filter.difficulty;
  const timeRange = filter.time;
  const type = filter.type;
  const flagName = (name !== null && name !== undefined && name !== '');
  let gamesAux=null;
  //Lo primero empezamos con la duracion del juego por lo bajo
  let games = state.items.filter(game => game.timePlay >= timeRange[0]);
  games = games.filter(game => game.timePlay <= timeRange[1]);

  //Filtrar por nombre
  if(flagName)
   games = games.filter(game => game.name.match(new RegExp(name, 'i')));

  //Primero por dificultad dificultad
  for(let d of dificulty)
   if(d.value !== undefined && d.value){
    if (gamesAux === null) gamesAux = [];
    for(let game of games)
     if(game.difficulty.match(new RegExp(d.name)))
      gamesAux.push(game)
   }

  //Segundo por tipo
  if(type.length > 0){
   games = (gamesAux === null) ? games: gamesAux;
   for(let t of type)
    if(t.value !== undefined && t.value){
     gamesAux = [];
     for(let game of games)
      if(game.type)
       if(game.type.match(new RegExp(t.name)))
        gamesAux.push(game)
    }
  }

  if(gamesAux===null) gamesAux = games;
  return gamesAux
 },
```

```
};
export const actions = {
    ...
 async loadGames({commit}){
  commit('setLoading', true);
  try{
   const games = [];
   const gamesDB = await db.collection(GAMES).get();
   gamesDB.forEach(game =>
    games.push({
     id: game.id,
     [NAME]: game.data()[NAME],
     [MIN_PLAYERS]: game.data()[MIN_PLAYERS],
     [MAX_PLAYERS]: game.data()[MAX_PLAYERS],
     [TYPE]: game.data()[TYPE],
     [DIFF]: game.data()[DIFF],
     [TIME]: game.data()[TIME],
     [IMG]: game.data()[IMG],
     [GAME_DESC]: game.data()[GAME_DESC],
     rate: 0,
     creatorId: game.data().creatorId
    })
   );
   commit('setLoadGames', games);
   commit('setLoading', false)
  } catch (e) {
   console.log(e)
   commit('setLoading', false)
  }
 },
 ....
};

export const mutations = {
    ...
 setLoadGames: (state, payload) => state.items = payload,
 setGameProfile: (state, payload) => state.gameProfile = payload,
    ...
};
```

6.12. Implementación de la seguridad

6.12.1. Introducción

Esta sección del anexo explicara brevemente que implementaciones se han realizado para mantener las reglas de seguridad de acceso a la aplicación. Se mantienen dos puntos de vista, parecida a la arquitectura de diseño del proyecto. Desde un punto de nivel de usuario limitaran los accesos desde la vista y por otra parte, la configuracion de la base de datos en el proveedor.

6.12.2. Implementacion en frontend

Es de vital importancia que la interfaz no de acceso a usuarios no registrados, de esta forma no se podra dar acceso a funcionalidades restringidas a usuarios que ya se encuentran en el sistema.

Para ello, se ha hecho una implementación en dos sentidos. Para los usuarios no registrados, si tratan de entrar en una seccion de la aplicacion para usuairos registrados, seran reenviados a la pagina principal de ShareBoard. Este código se ha implementado en el componente 'user_logged' del layout.

```
computed: {
    ...
    userLogged(){
        if(this.getUser === undefined || this.getUser ===  null)
            this.onLogout()
    },
    ...
}
...
methods: {
    onLogout(){
        this.$store.dispatch('users/logout');
        this.$router.push('/')
    },
}
```

Por otra parte, para mantener una mejor experiencia y evitar que cada vez que se usa la aplicación y ya nos encontremos registrados, se redirigirá a los usuairos registrados al menú principal. Este código se ha implementado en el componente 'default' del layout

```
computed: {
    ...
    userLogged() {
        if (this.getUser !== undefined && this.getUser !== null && !this.getNewUser){
            this.$router.push('/main/')
        }
            return false
```

```
        },
    ...
}
```

6.12.3. Implementación en backend

La base de datos que provee Firebase, Firebase Firestore, permite la configuración de reglas de seguridad para restringir el acceso a la base de datos. Esto servirá como segunda fase de seguridad, si en un primer lugar fallara o se rompiera la seguridad en el front-end.

La configuración de la base de datos no debe permitir que un usuario no registrado lea la base de datos y tampoco la modifique. Aludiendo a los requisitos funcionales, tampoco se debe permitir que un usuario que no es creador modifique un elemento, juego o reunión, que no le pertenece.

```
service cloud.firestore {
  match /databases/{database}/documents {
    //Acceso solo con registro
    match /{document=**} {
      allow read: if request.auth.uid != null;
    }
    //Modificar perfil solamente el propio usuario
    match  /users/{id} {
      allow update: if request.auth.uid == id;
    }
    //Modificar juego solamente el usuario creador
    match /games/{games}/creatorId/{id}{
      allow update: if request.auth.uid == id;
    }
    //Modificar reunion solamente el usuario creador
    match /meetups/{meetups}/creatorId/{id}{
      allow update: if request.auth.uid == id;
    }
  }
}
```

Figura 6.9: Definición de reglas de acceso a la base de datos de Firebase

6.13. Resultado de las pruebas de usuario

6.13.1. Introducción

Esta parte del anexo recoge los resultados de las pruebas de usuario . Se han realizado con un grupo de 8 personas en las que se encuentran estudiantes de informática como programadores y personas sin conocimientos de informática. Esto servirá par obtener distintas perspectivas y experiencias durante su interacción con la aplicación.

Se puede decir que las pruebas son de versión beta, ya que se esta exponiendo el sistema a usuarios fuera del entorno de desarrollo pero que cuentan a su disposición con el desarrollador del proyecto.

6.13.2. Objetivo y criterios de aceptación

El objetivo de estas pruebas es evaluar el estado del sistema y revisar que se cumplen los objetivos mínimos sin problemas de la aplicación. Para recoger la información que pueda ayudar a concluir sobre los dos objetivos descritos, se ha preparado un test que los usuarios rellenaran evaluando cada punto establecido en el test junto a una descripción para poder añadir cualquier otra valoración o apreciación.

En cada punto del test trata de focalizarse en un conjunto de requisitos licitados durante el análisis. A continuación se describe que parte pretende abarcar cada pregunta:

Usuario

1.1 Crear usuario: Recoge las funcionalidades de crear un usuario y posteriormente acceder a la aplicación.

1.2 Modificar perfil de usuario: Comprobar que el usuario no encuentra problema a la hora de manipular su perfil.

1.3: Gestión de amigos: Se comprueba el funcionamiento de añadir y eliminar amigos, pero junto a esto, también se evalúa la búsqueda de otros usuarios junto a los filtros.

Juegos

Crear juegos (2.1) modificar juego y buscar juegos (2.3) son los requisitos fundamentales para la interacción con los juegos. Para la busqueda sucede lo mismo qeu con los usaurios, es necesario tener en cuenta los filtros

Reuniones

Los puntos crear juego (3.1), modificar reunión (3.2) y búsqueda de reunión (3.3) son los requisitos parecidos a los dos puntos anteriores. Las siguientes comprobaciones son añadir (3.4) y eliminar usuarios (3.5) de una reunión. En esta parte es mas fácil encontrar errores en la búsqueda ya que el componente de filtro para reuniones tiene mas opciones. Para los requisitos creación(3.6) y visualización(3.7) están orientados al comportamiento de la clase evaluación.

Finalmente se evalúan las notificaciones (4.1) y la experiencia global del usuario (4.2). A continuacion el documento que ha servido como test descrito anteriormente.

Formulario de pruebas de usuario de ShareBoard

Tester ID	ID_TESTER
Plataforma: Android o Web Dispositivo:	Movil: Marca y modelo Web: Navegador

Nota: El campo “Experiencia” espera una valoración numérica. Las valoraciones se realizan del 1 al 5 sirven para puntuar la dificultad y lo intuible que ha sido la acción dentro de la aplicación siendo 1 como una mala experiencia y difícil, y un 5 como una buena fácil e intuitiva. Si no es suficiente con la puntuación, se puede utilizar el campo de “Descripción” para explicaciones con más detalle.

1 . Usuarios

1.1 Crear usuario:

- Registrarse como usuario, anotar en la descripción el método de registro, bien con cuenta de Google o registro en la pagina de ShareBoard.

Experiencia: ☐ Descripción:

1.2 Modificar perfil de usuario.

- Cambiar características del usuario y propiedades relacionadas con la interacción con la aplicación.

Experiencia: ☐ Descripción:

1.3 Gestión de amigos

- Añade y elimina amigos a tu lista de amigos.

Experiencia: ☐ Descripción:

2 . Juegos

2.1 Crear juego:

- Crear juego desde el principio.

Experiencia: ☐ Descripción:

2.2 Modificar juego:

- Modificar los valores del juego creado anteriormente.

Experiencia: ☐ Descripción:

2.3 Búsqueda de juegos

- Prueba a buscar un juego utilizando el filtro.

Experiencia: ☐ Descripción:

3 . Reuniones

3.1 Creación de reunión

- Crea una reunión desde el principio.

Experiencia: ☐ Descripción:

3.2 Modificar reunión

- Cambia los valores de una reunión.

Experiencia: ☐ Descripción:

3.3 Búsqueda de una reunión

- Utiliza el filtro de búsqueda para buscar una reunión que te interese.

Experiencia: ☐ Descripción:

3.4 Añadir usuarios a la reunión:

- Añade usuarios a tu reunión, sean amigos o no. Con distintos tipos de grupos (abiertos o cerrados, privados o públicos).

Experiencia: ☐ Descripción:

3.5 Eliminar usuarios de la reunión:

-Elimina usuarios de la reunión.

Experiencia: ☐ Descripción:

3.6 Evaluación de reunión:

-Evalúa el resultado de una reunión ya acontecida.

Experiencia: ☐ Descripción:

3.7 Visualización de evaluaciones.

- Visualiza el resultado de una evaluación

Experiencia: ☐ Descripción:

4 . Notificaciones

4.1 Notificaciones:

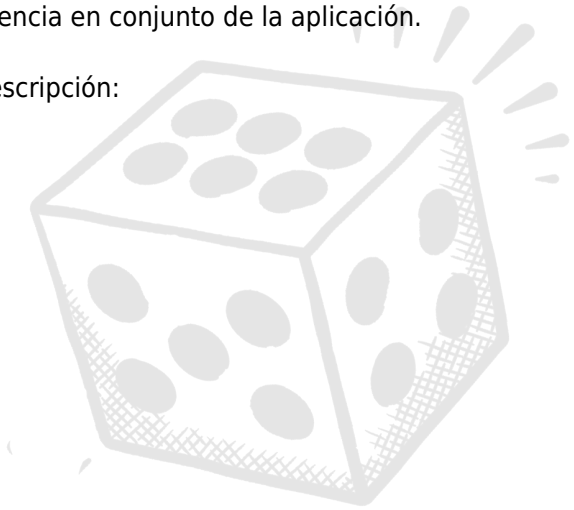
- Puntuá si la aplicación te ha notificado correctamente cuando ha sucedido un evento al que estabas registrado. También valora te hubiera gustado más notificaciones o de otro tipo

Experiencia: ☐ Descripción:

5. General:

- Valora al experiencia en conjunto de la aplicación.

Experiencia: ☐ Descripción:



6.13.3. Realización

Acceso a la aplicación

El acceso a la aplicación se ha realizado de dos formas distintas, una por cada plataforma para la que se ha desarrollado la aplicación. Para los usuarios de la web, se les ha proporcionado una dirección al hosting de Firebase de la aplicación web. Por otro lado, a los probadores de la aplicación móvil en Android, se les ha facilitado el archivo de instalacion APK (Android Application Package) a través del correo electrónico.

En común a ambas plataformas, los probadores pueden iniciar el proceso de registro de a la aplicación y posteriormente acceder a ella.

Duración

Las pruebas tendrán una duración de dos semanas con el fin de que se puedan crear reuniones y se pueda crear una interacción real con otros usuarios. En este tiempo los usuarios podrán buscar errores como simular el uso de la aplicación.

6.13.4. Resultados

Los resultados serán descritos en en función de las apreciaciones, tanto positivas como negativas, aportadas por los probadores. La evaluación se realiza de dos formas, una con las apreciaciones y otra con el valor numérico dado para cada parte de la aplicación. Comenzaremos por los valores

1.1 Crear usuario: Varios usuarios a solicitado que sería necesario una confirmación de email, sobre todo para facilitar la compresión de la correcta creación de la cuenta. El resto de aspectos, como el formulario o el registro con cuenta de Google a sido recogidos con satisfacción.

1.2 Modificar perfil de usuario: La presentación de los datos ha sido de buen gusto para los probadores. Se ha echado en falta que al visitar el perfil de otro amigo, aparezcan los amigos en común con el otro usuario.

1.3 Gestión de amigos La interacción en esta sección a recibido mas criticas en su interacción. Algunos usuarios esperan que al eliminar a un amigo, sea la aplicación quien te lleva de nuevo a tu perfil de usuario. Más urgente es que algunos usuarios han encontrado dificultades en como añadir amigos. Estos mismos han sugerido añadir un enlace directo, como sucede con los juegos o las reuniones, para ver al resto de usuarios.

2.1 Crear juego: Los probadores no han encontrado ningún aspecto negativo o mejorable en esta sección.

2.2 Modificar juego: En esta sección se ha encontrado un requisito fundamental que no ha sido diseñado ni implementado, eliminar juego de la base de datos. Seria importante implementarlo.

2.3 Búsqueda de juegos: En general ha sido satisfactorio, exceptuando que los usuarios no pueden filtrar o ver los juegos que el mismo ha creado, de tal forma, que se dificulta el encontrarlos para modificar los juegos.

3.1 Creación de reunión: El formato de fecha introducido (mm/dd/aaaa) no es intuitivo para los usuarios. No ha impedido crear reuniones pero resulta muy confuso para los usuarios.

3.2 Modificar reunión: Hay ciertos valores de las reuniones que se ha obviado su modificación para garantizar mas estabilidad. Los probadores no se han encontrado a gusto con esta característica, han pedido que resultaría mas fácil si se pudieran gestionar los juegos, la fecha y la hora una vez creada la reunión.

3.3 Búsqueda de reunión:EL filtro de distancia de reunión permite distancia mínima de 1 Km. Esto evita encontrar las reuniones mas próximas y de mayor interés del usuario. El valor mínimo del filtro de distancia es 1, por lo tanto no permite encontrar reuniones cercanas.

3.4 Añadir usuario a la reunión: Ha resultado cómodo para la mayoría de probadores. Algunos han sugerido que se puedan añadir usuarios directamente desde la creación de la reunión.

3.5 Eliminar usuario a la reunión: En general ha sido aceptada positivamente este requisito, aunque algunos probadores han sugerido que solicite confirmación de la aplicación para eliminar un usuario de la reunión.

3.6 Evaluación de la reunión Los probadores no han encontrado de buen gusto que un usuario se evalué a si mismo.

3.7 Visualización de la reunión Los probadores no han encontrado ningún aspecto negativo o mejorable en esta sección.

4 Notificaciones: En este aspecto los probadores han quedado satisfechos. Un probador sugirió que estas notificaciones fueran, o pudieran establecerse como silenciosas como mejora de la interacción.

El grado de satisfacibilidad se interpretara en función de la valoración cuantificada de cada requisitos explorado. La figura 6.10 muestra la media obtenida en cada punto, tanto en la versión web como Android.

Se puede observar en la gráfica que los requisitos han sido satisfactorios en la su media, ya que todos los valores están por encima de la mitad, por encima de 2,5 puntos. Se puede observar que en cada apartado, los requisitos menos valorados han sido la búsqueda de juegos, usuarios y reuniones,

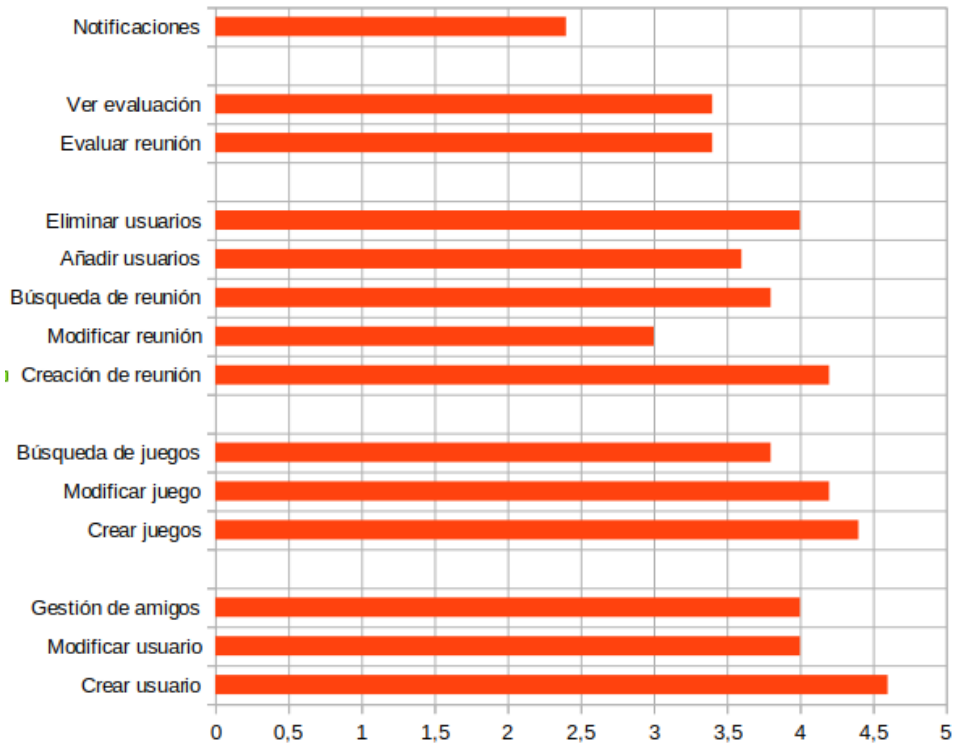


Figura 6.11: Gráfico de barras de satisfacción de usuarios por requisito.

esto puede decir que tal vez la disposición de los filtros y su efectividad no han sido claras, como por ejemplo en la búsqueda de reuniones, un filtro de distancia que no permite buscar reuniones cercanas.

6.13.5. Conclusión

En general a los probadores les ha resultado muy intuitiva el uso de la aplicación. La interfaz ha sido encontrada de buen gusto, pero si que se han encontrado problemas generales como la facilidad de los filtros. En conjunto los filtros no han resultado muy eficientes por dos razones, porque no era intuitivo usarlos sin un botón de aplicar y que ocupaban demasiado espacio en la interfaz cuando son desplegados.

Otra de las características que parecen tener sus defectos es la modificación de reuniones. Por otro parte, las reuniones no han parecido tener la mejor acogida tanto en su visualización como en su creación, han sido las evaluaciones de reuniones. El único requisitos que no parece haber superado la mitad son las notificaciones. Esto se debe a que la mayoría de probadores realizaron sus pruebas sobre la versión Web, que no tenia implementadas las notificaciones en segundo plano.

Respecto a la evaluación cuantificada obtenida, aprueba la aplicación en su fase beta. En otro aspectos mas cualitativos, opiniones de los probadores, seria necesario realizar algunos cambios para

continuar con las siguientes iteraciones y no continuar arrastrando errores, o procrastinando soluciones. Restablecer la posición de los filtros y mejorar la integración de las notificaciones serian los dos puntos mas importantes para acabar la cuarta iteración y continuar con el desarrollo.